

**Name:** Ms. Prarthi Hrishit Kothari

**Class:** M. Sc. Bioinformatics (Part I)

**Roll Number:** 115

**Course:** M. Sc. Bioinformatics

**Department:** Department of Bioinformatics

**Paper:** Mandatory Paper II

**Paper Name  
and Code:** Java Programming, Introduction  
to Linux and Machine Learning  
(GNKPSBI2P502)

**Academic  
Year:** 2023-24



**SGCP's**  
**Guru Nanak Khalsa College**  
**of Arts, Science & Commerce (Autonomous)**

## **DEPARTMENT OF BIOINFORMATICS**

### **CERTIFICATE**

This is to certify that Ms. Prarthi Hrishit Kothari ( Roll No: 115 ) of M. Sc. Bioinformatics (Part I) has satisfactorily completed the practical for Mandatory Paper II: Java Programming, Introduction to Linux and Machine Learning (GNKPSBI2P502) for Semester II course as prescribed by the University of Mumbai during the academic year 2023-2024.

---

Teacher-in-Charge  
(Signature)

---

Head of Department  
(Signature)

---

External Examiner  
(Signature)

# INDEX

[illegible]

**Date: 04/02/2024**

## **Practical 1**

### **JAVA: Basic Programming**

#### **AIM:**

To demonstrate the basic operations and functionalities of Java programming language.

#### **INTRODUCTION:**

Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform in itself. It is a widely-used programming language and computing platform that was first released by Sun Microsystems in 1995. It is a high-level, class-based, object-oriented language designed to have as few implementation dependencies as possible. Java is known for its portability, as compiled Java code can run on all platforms that support Java, without the need to recompile.

To create an application using Java, one requires the Java Development Kit (JDK) to be downloaded, which is available for Windows, macOS, and Linux. A program written in the Java programming language is converted into Java bytecode by the compiler, which runs without modification on any system that supports the Java Virtual Machine (JVM). The Java software platform consists of the JVM, the Java API, and a complete development environment.

#### **ADVANTAGES OF JAVA –**

- 1. Object-Oriented:** Java is an object-oriented language, which facilitates the development of modular programs and the reuse of code. This flexibility and reusability increase productivity and facilitate the development of large-scale applications.
- 2. Portability:** The "write once, run anywhere" (WORA) approach of Java allows code to be built once and executed on any platform that supports Java without requiring recompilation, making it platform-independent and enabling the same code to operate on several systems without modification.
- 3. Robustness and Reliability:** Java is a robust and reliable language, as it includes strong memory management, exception handling, and type safety features.
- 4. Security:** Java is known for its built-in security features, which help protect systems from harmful viruses and malware. It provides a secure environment for running applications, particularly in networked environments.
- 5. Community Support and Open Source:** Java is an open-source language with a large and active community. This means that developers have access to a wide range of libraries, frameworks, and tools, as well as a wealth of online resources and support.

### **DISADVANTAGES OF JAVA –**

- 1. Memory Usage:** Java programs run on top of the Java Virtual Machine (JVM), which consumes more memory than other programming languages.
- 2. Slower Execution Speed:** Java's platform independence comes at a cost to performance, as Java programs take longer to run compared to C/C++ programs.
- 3. Lack of Multiple Inheritance:** Java does not support multiple inheritance, which can make it more difficult to reuse code and create complex class hierarchies.
- 4. Syntax Verbosity:** Java's syntax is more verbose than other programming languages, such as Python, which can make the code cumbersome and more challenging to read.
- 5. Low-Level Access:** Java does not support low-level programming, such as pointers, which can limit its use in certain applications.

### **APPLICATIONS OF JAVA –**

- 1. Graphical User Interfaces (GUI):** Java is used to develop desktop applications, through APIs such as AWT (Abstract Windowing Toolkit) and Swing, due to its object-oriented nature and platform independence.
- 2. Mobile Applications:** Java is the official programming language for Android, making it a popular choice for developing mobile applications. Java's portability and low-maintenance capabilities make it suitable for creating high-performance mobile apps.
- 3. Web Applications and Services and Cloud Computing:** Java is widely used for developing web applications and services, such as web servers, servlets, and JSPs, and for developing cloud computing applications, such as cloud servers and cloud-based services.
- 4. Scientific Applications:** Java is used for developing scientific applications, such as mathematical simulations and data analysis owing to its performance and scalability.
- 5. Enterprise Solutions:** Java is used for developing enterprise solutions, such as banking applications and supply chain management systems.

## DATATYPES –

A datatype determines the amount of memory allocated and the type of the value that will be stored in the variable. In Java, there are two categories of data types – primitive data types and non-primitive data types.

1. **Primitive data types** – Also called as Intrinsic data types, these include byte, short, int, long, float, double, boolean, and char.

Data Types	Size (Bytes)	Description	Range	Default value
byte	1	Stores whole numbers	-128 to 127	0
short	2	Stores whole numbers	-32,768 to 32,767	0
int	4	Stores whole numbers	-2,147,483,648 to 2,147,483,647	0
long	8	Stores whole numbers	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
float	4	Stores fractional numbers.	3.4E+038 to 3.4E-038 (Precision – 6 to 7 digits)	0.0f
double	8	Stores fractional numbers.	1.7E+308 to 1.7E-308 (Precision – up to 15 digits)	0.0
char	2	Stores a single character/letter or ASCII values	-	Null
boolean	1	Stores true or false values	true or false	false

2. **Non-primitive data types** – Also termed as Reference data types, these include String, Arrays, and Classes.

## OPERATORS –

Various types of operators include –

1. **Arithmetic Operators** –

Name of Operator	Symbol	Description	Syntax	Example (int a = 3, b = 2)
Addition	+	Returns the sum	int c = a + b	c = 5
Subtraction	-	Returns the difference	int c = a - b	c = 1
Multiplication	*	Returns the product	int c = a * b	c = 6
Division	/	Returns the quotient	int c = a / b	c = 1
Modulus	%	Returns the remainder	int c = a % b	c = 1

## 2. Relational Operators –

Name of Operator	Symbol	Description	Syntax	Example (int a = 3, b = 2)
Equal to (Comparison)	==	Compares the operand on the L.H.S. to the operand on the R.H.S. and checks if they are equal to each other.	a == b	It will check if (3 == 2) and return false.
Not Equal to	!=	Inequality operator – Compares the operand on the L.H.S. to the operand on the R.H.S. and checks if they are not equal to each other.	a != b	It will check if (3 != 2) and return true.
Greater than	>	Checks if the operand on the L.H.S. is greater than the operand on the R.H.S.	a > b	It will check if (3 > 2) and return true.
Less than	<	Checks if the operand on the L.H.S. is less than the operand on the R.H.S.	a < b	It will check if (3 < 2) and return false.
Greater than or equal to	>=	Checks if the operand on the L.H.S. is greater than or equal to the operand on the R.H.S.	a >= b	It will check if (3 >= 2) and return true.
Less than or equal to	<=	Checks if the operand on the L.H.S. is less than or equal to the operand on the R.H.S.	a <= b	It will check if (3 <= 2) and return false.

## 3. Logical Operators –

Name of Operator	Symbol	Description	Syntax	Example (int a = 3, b = 2, c = 1)
Logical AND	&&	Joins 2 expressions in conjunction i.e., returns true if both the conditions are true or not	a > b && a > c	It will check if (3 > 2 && 3 > 1) and return true since 3 is greater than both 2 and 1.
Logical OR		Joins 2 expressions in disjunction i.e., 1. returns true if both the conditions / any one of the conditions is true or not 2. returns false if both the conditions are false.	a > b    a < c	It will check if (3 > 2    3 < 1) and return true since one of the conditions i.e., 3 is greater than 2 returns true.
Logical NOT	!	Negates an expression	!(a > b)	!(3 > 2): It will return false since it negates the condition that 3 is greater than 2, which is true.

#### 4. Assignment Operators / Shorthand Operators –

Name of Operator	Symbol	Description	Syntax	Example (int a = 3, b = 2)
Equal to	=	Assigns the value on the R.H.S. to the variable on the L.H.S.	a = b	a = 2 Assigns the value of b to a.
Addition – Equal to	+=	Assigns the sum of both the operands to the operand on the L.H.S.	a +=b	(a = a + b) a = 3 + 2 a = 5
Subtraction – Equal to	-=	Assigns the difference between the operand on the L.H.S and the operand on the R.H.S. to the operand on the L.H.S.	a-=b	(a = a - b) a = 3 – 2 a = 1
Multiplication – Equal to	*=	Assigns the product of both the operands to the operand on the L.H.S.	a*=b	(a = a * b) a = 3 * 2 a = 6
Division – Equal to	/=	Assigns the quotient of the division of both the operands to the operand on the L.H.S.	a/=b	(a = a / b) a = 3 / 2 a = 1
Modulus – Equal to	%=	Assigns the remainder of the division of both the operands to the operand on the L.H.S.	a%=b	(a = a % b) a = 3 % 2 a = 1

#### 5. Increment / Decrement Operators –

Name of Operator	Symbol	Description	Syntax	Example (int a = 3)
Increment	++	Increases the value of the operand by 1.	a++	a = 3 a = 3 + 1 a = 4 Value of a increases by 1.
Decrement	--	Decreases the value of the operand by 1.	a--	a = 3 a = 3 – 1 a = 2 Value of a decreases by 1.



## PACKAGES –

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces, which can be accessed using the keyword **import**. Packages are used for –

1. Preventing naming conflicts
2. Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
3. **Providing controlled access** – protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
4. Packages can be considered as **data encapsulation (or data-hiding)**.

### Types of packages –

1. **Built-in Packages** – These packages consist of a large number of classes which are a part of Java API. Some of the commonly used built-in packages are –
  - (a) **java.lang** – Contains language support classes (example – classes which define primitive data types, math operations). This package is automatically imported.
  - (b) **java.io** – Contains classes for supporting input / output operations.
  - (c) **java.util** – Contains utility classes which implement data structures like Linked List, Dictionary, Scanner class and support for Date / Time operations.
  - (d) **java.applet** – Contains classes for creating Applets.
  - (e) **java.awt** – Contain classes for implementing the components for graphical user interfaces (like button, menus etc).
2. **User-defined packages** – These are the packages that are defined by the user.

### Syntax for Accessing classes inside a package –

Syntax	Example	Description
import package.*;	import java.util.*;	All the classes and interfaces of this package will be accessible but not subpackages.
import package.class_name;	import java.util.Scanner;	Only mentioned class (Scanner class) of this package will be accessible.

## CLASSES AND OBJECTS –

A **class** is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. The keyword **class** is used for declaration of the class. Classes are required in OOPs because –

1. It provides template for creating objects, which can bind code into data.
2. It has definitions of methods and data.
3. It supports inheritance property of Object – Oriented Programming and hence can maintain class hierarchy.
4. It helps in maintaining the access specifications of member variables.

**Objects** are the basic unit of Object – Oriented Programming and it represent the real – life entities. An object consists of –

1. **State** – Represented by attribute, it shows properties of an object.
2. **Behavior** – Represented by methods, it shows response of an object with other objects.
3. **Identity** – It gives a unique name to an object. It also grants permission to one object to interact with other objects.

Objects are required in OOPs because they can be created to call non – static functions which are not present inside the **main** method but present inside the Class and also provide the name to the space which is being used to store the data.

### Syntax for creation of an Object of the Class –

```
Class_name Object_name = new Class_name(parameters_if_any);  
// keyword new is used to allocate memory for the object created
```

## SCANNER CLASS –

In Java, **Scanner** is a class in **java.util** package used for obtaining the input of the primitive types like int, double, etc. and strings.

Method	Description
nextByte()	Used to read a <b>byte</b> value from the user
nextShort()	Used to read a <b>short</b> value from the user
nextInt()	Used to read an <b>integer</b> value from the user
nextLong()	Used to read a <b>long</b> value from the user
nextFloat()	Used to read a <b>float</b> value from the user
nextDouble()	Used to read a <b>double</b> value from the user
next() or nextChar()	Used to read a <b>single character</b> value from the user
nextBoolean()	Used to read a <b>boolean</b> value from the user
nextLine()	Used to read a <b>String</b> value from the user

## **PROGRAMS –**

### **Question 1 –**

Write a java program to create a class named 'Calculate' and find the area and the perimeter of a rectangle.

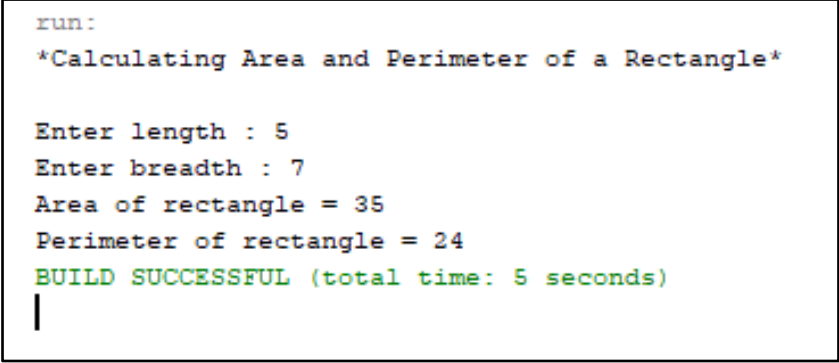
### **CODE –**

```
package Practs1;
import java.util.*;
import java.io.*;

public class Calculate {
    int area, peri;
    public int area(int l, int b)
    {
        area = l*b;
        return area;
    }
    public int perimeter(int l, int b)
    {
        peri = 2*(l+b);
        return peri;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("*Calculating Area and Perimeter of a Rectangle*");
        int length, breadth;
        System.out.print("\nEnter length : ");
        length = sc.nextInt();
        System.out.print("Enter breadth : ");
        breadth = sc.nextInt();
        Calculate obj = new Calculate();
        System.out.println("Area of rectangle = " + obj.area(length,breadth));
        System.out.println("Perimeter of rectangle = " + obj.perimeter(length,breadth));
    }
}
```

### **OUTPUT –**



```
run:
*Calculating Area and Perimeter of a Rectangle*

Enter length : 5
Enter breadth : 7
Area of rectangle = 35
Perimeter of rectangle = 24
BUILD SUCCESSFUL (total time: 5 seconds)
|
```

**Figure 1:** Calculating and displaying the area and perimeter of a rectangle

### Question 2 –

Create a class named 'Student' with String variable 'name' and integer variable 'roll\_no'. Through user input, assign the value of 'roll\_no' as '2' and that of 'name' as "John" by creating an object of the class 'Student'.

### CODE –

```
package Practs1;
import java.util.*;
import java.io.*;

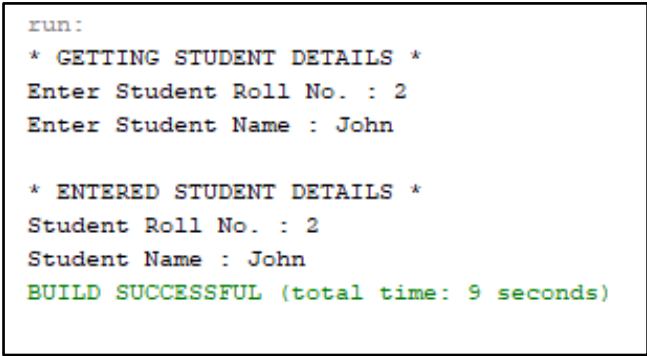
public class Student {

    Scanner sc = new Scanner(System.in);
    String name;
    int roll_no;

    public void getDetails()
    {
        System.out.println("* GETTING STUDENT DETAILS *");
        System.out.print("Enter Student Roll No. : ");
        roll_no = sc.nextInt();
        System.out.print("Enter Student Name : ");
        name = sc.next();
        System.out.println("\n* ENTERED STUDENT DETAILS *");
        System.out.println("Student Roll No. : " + roll_no);
        System.out.println("Student Name : " + name);
    }

    public static void main(String[] args) {
        Student obj = new Student();
        obj.getDetails();
    }
}
```

### OUTPUT –



```
run:
* GETTING STUDENT DETAILS *
Enter Student Roll No. : 2
Enter Student Name : John

* ENTERED STUDENT DETAILS *
Student Roll No. : 2
Student Name : John
BUILD SUCCESSFUL (total time: 9 seconds)
```

**Figure 2:** Retrieving and displaying the details of a student

**Question 3 –**

Write a program to print the area of two rectangles having sides (4,5) and (5,8) respectively by creating a class named 'Rectangle' with a method named 'Area' which returns the area with length and breadth passed as parameters (obtained through user input) to its constructor.

**CODE –**

```
package Practs1;
import java.util.*;
import java.io.*;

public class Rectangle {

    int length, breadth, area;

    Rectangle()
    {
        System.out.println("* AREA OF RECTANGLE * \n");
    }

    Rectangle(int l, int b)
    {
        length = l;
        breadth = b;
    }

    public int Area()
    {
        area = length*breadth;
        return area;
    }

    public static void main(String[] args) {
        int length, breadth;
        Rectangle r1 = new Rectangle(); //default constructor
        Scanner sc = new Scanner(System.in);

        // run1
        System.out.println("* RUN 1 *");
        System.out.print("Enter length : ");
        length = sc.nextInt();
        System.out.print("Enter breadth : ");
        breadth = sc.nextInt();

        Rectangle r2 = new Rectangle(length, breadth);
```

```

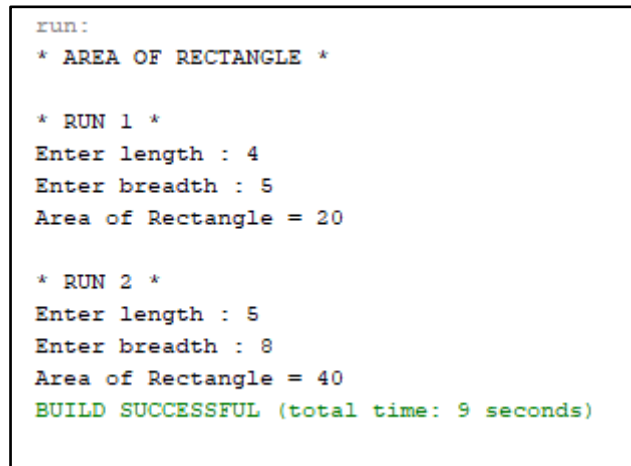
        System.out.println("Area of Rectangle = " + r2.Area());

//run 2
System.out.println("\n* RUN 2 *");
System.out.print("Enter length : ");
length = sc.nextInt();
System.out.print("Enter breadth : ");
breadth = sc.nextInt();

Rectangle r3 = new Rectangle(length, breadth);
System.out.println("Area of Rectangle = " + r3.Area());
    }
}

```

### OUTPUT –



```

run:
* AREA OF RECTANGLE *

* RUN 1 *
Enter length : 4
Enter breadth : 5
Area of Rectangle = 20

* RUN 2 *
Enter length : 5
Enter breadth : 8
Area of Rectangle = 40
BUILD SUCCESSFUL (total time: 9 seconds)

```

**Figure 3:** Calculating and displaying the area of two rectangles by passing length and breadth as parameters to its constructor

#### Question 4 –

Print the average of three numbers entered by the user by creating a class named 'Average' having a method to calculate and print the average.

#### CODE –

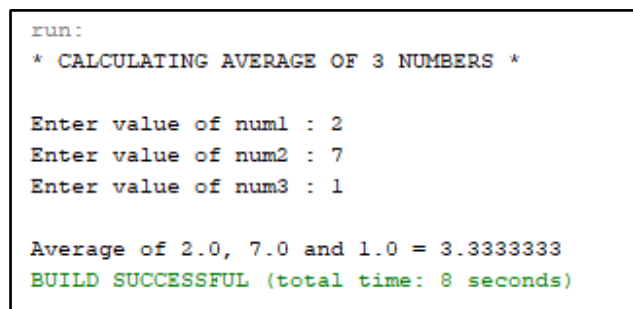
```
package Practs1;
import java.util.*;
import java.io.*;

public class Average {
    Scanner sc = new Scanner(System.in);
    float num1, num2, num3;
    float average;

    public void avg()
    {
        System.out.print("\nEnter value of num1 : ");
        num1 = sc.nextFloat();
        System.out.print("Enter value of num2 : ");
        num2 = sc.nextFloat();
        System.out.print("Enter value of num3 : ");
        num3 = sc.nextFloat();
        average = (num1 + num2 + num3)/3;
        System.out.println("\nAverage of " + num1 + ", " + num2 + " and " + num3 + " = " + average);
    }

    public static void main(String[] args) {
        System.out.println("* CALCULATING AVERAGE OF 3 NUMBERS *");
        Average obj = new Average();
        obj.avg();
    }
}
```

#### OUTPUT –



```
run:
* CALCULATING AVERAGE OF 3 NUMBERS *

Enter value of num1 : 2
Enter value of num2 : 7
Enter value of num3 : 1

Average of 2.0, 7.0 and 1.0 = 3.3333333
BUILD SUCCESSFUL (total time: 8 seconds)
```

**Figure 4:** Calculating and displaying the average of three user – entered numbers

**Question 5 –**

Write a program that would print the information (name, year of joining, address) of three employees by creating a class named 'Employee'. The output should be as follows:

Name	Year of Joining	Address
Robert	1994	64C – WallStreet
Sam	2000	68D – WallStreet
John	1999	26B – WallStreet

**CODE –**

```
package Practs1;
import java.util.*;
import java.io.*;

public class Employee {
    Scanner sc = new Scanner(System.in);
    String name[] = new String[3];
    int year_of_joining[] = new int[3];
    String address[] = new String[3];

    public void Details()
    {
        int i;
        for(i=0 ; i<3 ; i++)
        {
            System.out.println("\n* DETAILS OF EMPLOYEE " + (i+1) + " *");
            System.out.print("Enter Name : ");
            name[i] = sc.next();
            System.out.print("Enter Year of Joining : ");
            year_of_joining[i] = sc.nextInt();
            System.out.print("Enter Address : ");
            address[i] = sc.next();
        }

        System.out.println("\n*****");
        System.out.println("* TABLE - DETAILS OF EMPLOYEE *");
        System.out.println("*****");
        System.out.println("\nNAME\t YEAR OF JOINING\t ADDRESS");
        System.out.println("-----");
        for(i=0 ; i<3 ; i++)
        {
            System.out.println(name[i] + "\t\t" + year_of_joining[i] + "\t\t" + address[i] +
            "\t");
        }
    }
}
```



```

public static void main(String[] args) {
    System.out.println("* EMPLOYEE DETAILS *");
    Employee obj = new Employee();
    obj.Details();
}
}

```

**OUTPUT –**

```

run:
* EMPLOYEE DETAILS *

* DETAILS OF EMPLOYEE 1 *
Enter Name : Robert
Enter Year of Joining : 1994
Enter Address : 64C-WallStreet

* DETAILS OF EMPLOYEE 2 *
Enter Name : Sam
Enter Year of Joining : 2000
Enter Address : 68D-WallStreet

* DETAILS OF EMPLOYEE 3 *
Enter Name : John
Enter Year of Joining : 1999
Enter Address : 26B-WallStreet

*****
* TABLE - DETAILS OF EMPLOYEE *
*****

NAME      YEAR OF JOINING      ADDRESS
-----
Robert      1994      64C-WallStreet
Sam          2000      68D-WallStreet
John         1999      26B-WallStreet
BUILD SUCCESSFUL (total time: 1 minute 37 seconds)
|

```

**Figure 5:** Retrieving and displaying the information of three employees with respect to their  
– Name, Year of Joining and Address

## **RESULTS:**

Using the Java programming language, the basic operations and functionalities of Java were demonstrated in order to retrieve user input for calculating and displaying the area and perimeter of a rectangle, retrieving and displaying the details of a student, calculating the area of a rectangle by passing length and breadth of the rectangle, obtained from the user, as parameters to its constructor, calculating and displaying the average of three user – entered numbers and retrieving and displaying the details of employees with respect to their name, year of joining and address.

## **CONCLUSION:**

The basic operations and functionalities of Java programming language were demonstrated.

## **REFERENCES:**

1. What is Java? - Java Programming Language Explained - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/what-is/java/>
  2. What is Java technology and why do I need it? (n.d.). [https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html)
  3. Payne, R. (2023, September 25). Advantages and Disadvantages of Java. Developer.com. <https://www.developer.com/java/advantages-and-disadvantages-of-java/>
  4. Top 10 Applications of Java in Real World. (2022, October 12). GeeksforGeeks. <https://www.geeksforgeeks.org/top-10-applications-of-java-in-real-world/>
  5. Java Data Types. (2023, December 13). GeeksforGeeks. <https://www.geeksforgeeks.org/data-types-in-java/>
  6. Operators in Java. (2023, July 20). GeeksforGeeks. <https://www.geeksforgeeks.org/operators-in-java/>
  7. Packages In Java. (2023, May 4). GeeksforGeeks. <https://www.geeksforgeeks.org/packages-in-java/>
  8. Scanner Class in Java. (2023, December 13). GeeksforGeeks. <https://www.geeksforgeeks.org/scanner-class-in-java/>
  9. Classes and Objects in Java. (2023, September 22). GeeksforGeeks. <https://www.geeksforgeeks.org/classes-objects-java/>
-

**Date – 05/02/2024**

## **Practical 2**

### **JAVA: Inheritance and Polymorphism**

#### **AIM:**

To explore and demonstrate the concept of inheritance and polymorphism in Java programming language.

#### **INTRODUCTION:**

Java is a versatile programming language released in 1995 that runs on various platforms without needing recompilation. Its object-oriented design and platform independence make it popular for diverse applications, from web development to mobile apps. Known for its "write once, run anywhere" capability, Java simplifies development and promotes code reusability across systems.

#### **INHERITANCE**

Within the object-oriented programming paradigm of Java, inheritance serves as a mechanism for the creation of new classes (subclasses) based upon existing classes (superclasses). This enables the subclass to inherit both the methods and attributes of the superclass, while also introducing its own unique methods and attributes as needed. The extends keyword is used for inheritance in Java. The following advantages justify as to why inheritance is required in Java programming language –

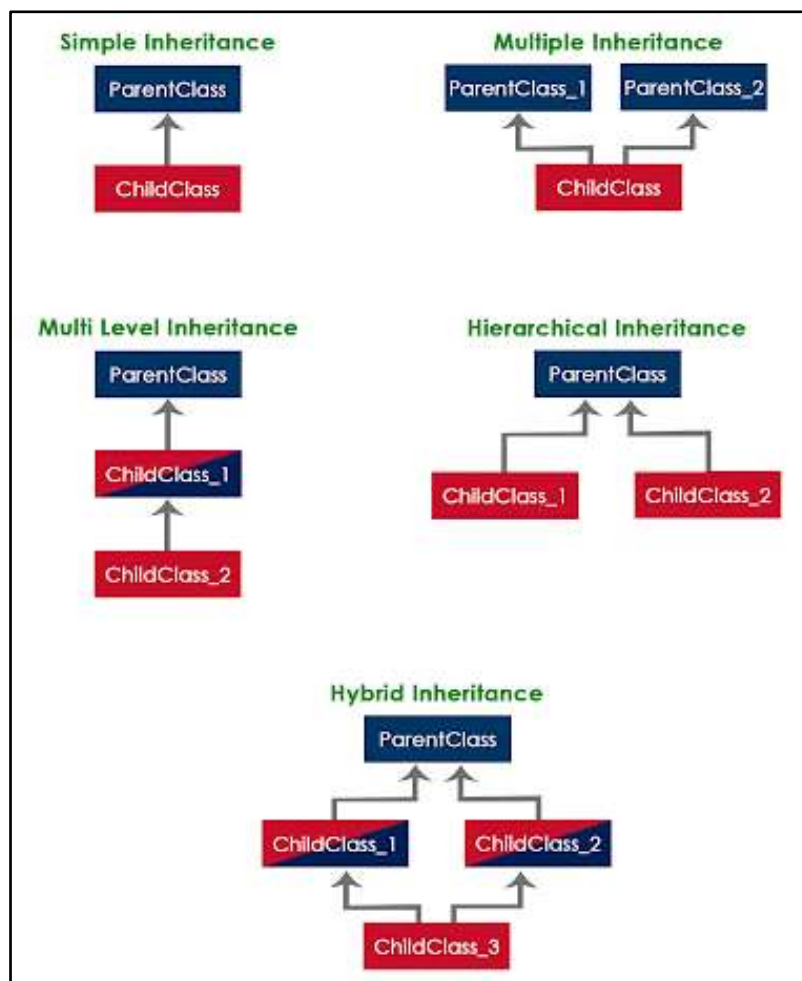
- 1. Enhanced Code Reusability** – The inheritance mechanism in Java facilitates the reuse of code defined in the parent class ("Superclass") by its descendant classes ("Subclasses"). This eliminates the need for redundant code, promoting efficiency and maintainability.
- 2. Enabling Method Overriding** – Inheritance serves as the foundation for achieving "Method Overriding" in Java. This concept allows subclasses to redefine inherited methods from the superclass, providing specialized implementations customized to their specific needs. This contributes to runtime polymorphism, whereby objects of different subclasses can respond differently to the same method call based on their actual type.
- 3. Abstraction through Inheritance** – Inheritance plays a crucial role in realizing the concept of "Abstraction" within Java. By allowing the definition of abstract classes and methods, inheritance enables the representation of core functionalities without exposing unnecessary implementation details to the user. This fosters a clearer understanding of the system's overall behavior while ensuring data and logic encapsulation.

## Types of Inheritance

Type of Inheritance	Description	Advantage	Syntax
Single Inheritance	A subclass (child class) is derived from only one superclass (parent class). It inherits the properties and behavior of a single parent class.	<ol style="list-style-type: none"> <li>1. Simple and easy to understand, avoiding ambiguity in method overriding.</li> <li>2. Promotes code reusability for common functionalities defined in the superclass.</li> </ol>	<pre> class Parent_Class {     //data members and functions of parent class }  class Parent_Class <b>extends</b> Child_class{     // data members and functions of child class } </pre>
Multilevel Inheritance	A subclass (child class) inherits from another subclass (intermediate class), which itself inherits from its own superclass (parent class), forming a chain of inheritance.	<ol style="list-style-type: none"> <li>1. Useful for modeling hierarchical relationships.</li> <li>2. Enables code reuse across multiple levels of the hierarchy.</li> </ol>	<pre> class Parent_Class {     //data members and functions of parent class }  class Intermediate_Class <b>extends</b> Parent_class{     // data members and functions of intermediate class }  class Child_Class <b>extends</b> Intermediate_class{     // data members and functions of child class } </pre>
Multiple Inheritance	Java does not directly support multiple inheritance (inheriting from multiple classes) due to potential method ambiguity issues (diamond problem). Hence, Interfaces are used to	<ol style="list-style-type: none"> <li>1. Provides flexibility to inherit functionalities from multiple unrelated classes but avoids diamond problems.</li> </ol>	<pre> interface Interface_one {     // declaration of abstract methods }  interface Interface_two {     // declaration of abstract methods }  class Child_Class <b>implements</b> Interface_one, Interface_two {     // defining the abstract methods of Interface_one and Interface_two along </pre>

	achieve a similar effect. A class can implement multiple interfaces, inheriting their abstract methods and properties.		with data members and functions of child class }
Hierarchical Inheritance	Multiple subclasses (child classes) inherit from the same superclass (parent class), creating a tree – like structure.	<ol style="list-style-type: none"> <li>1. Promotes code reusability for shared features among related classes.</li> <li>2. Simplifies code by avoiding code duplication in subclasses with common functionality.</li> </ol>	<pre> class Parent_Class {     //data members and functions of parent class }  class Child_Class1 <b>extends</b> Parent_class{     // data members and functions of child_class1 class }  class Child_Class2 <b>extends</b> Parent_class{     // data members and functions of child class2 } </pre>
Hybrid Inheritance	Hybrid inheritance is the combination of two or more types of inheritance. It is a combination of hierarchical and multiple inheritance.	<ol style="list-style-type: none"> <li>1. Promotes reusability and modularity of code.</li> </ol>	<pre> interface Interface_one {     // declaration of abstract methods }  interface Interface_two {     // declaration of abstract methods }  class Parent_Class {     //data members and functions of parent class }  class Intermediate_Class <b>extends</b> Parent_class{     // data members and functions of intermediate class } </pre>

			<pre> class Child_Class <b>extends</b> Intermediate_class <b>implements</b> Interface_one, Interface_two {      // defining the abstract methods of     Interface_one and Interface_two along     with data members and functions of     child class and intermediate class  } </pre>
--	--	--	---



**Figure 1:** Types of Inheritance

## **POLYMORPHISM**

In Java, polymorphism refers to an object's capacity for several forms. It is among the core ideas of programming that is object-oriented. Writing generic code that can handle objects of many types is made easier by polymorphism, which enables objects of distinct classes to be considered as objects of a common type. Java uses the principles of method overloading and overriding to implement polymorphism.

### **Types of Polymorphism**

#### **1. Compile – time Polymorphism / Static Polymorphism –**

Achieved through method overloading, it involves defining several methods with the same name but distinct signatures (parameters and return types). The method that the compiler selects at build time is determined on the types of arguments.

#### **Example –**

```
class CompileTimeExample {

    // Method with 2 integer parameters
    public int function_name_add(int a, int b)
    {
        // Returns sum of integer numbers
        return (a + b);
    }

    // Method 2 with same name but with 2 double parameters
    public double function_name_add(double a, double b, double c)
    {
        // Returns sum of double numbers
        return (a + b + c);
    }
}

public static void main(String[] args)
{
    CompileTimeExample obj = new CompileTimeExample( );
    // Calling method by passing input as in arguments
    System.out.println(obj.function_name_add(2, 4));
    System.out.println(obj.function_name_add(5.5, 6.3, 7.5));
}
```

## 2. Run – time Polymorphism / Dynamic Polymorphism –

Achieved through method overriding, it involves a mechanism that allows objects of different classes to be treated as if they belonged to the same class, enabling the execution of the correct method based on the object's actual type at runtime.

**Example –**

### **RunTimeExample1.java**

```
class RunTimeExample1 {  
  
    // Method with 2 integer parameters  
    public int function_name_add(int a, int b)  
    {  
        // Returns sum of integer numbers  
        return (a + b);  
    }  
}
```

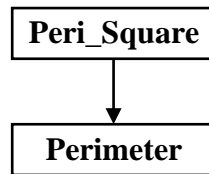
### **RunTimeExample2.java**

```
class RunTimeExample2 extends RunTimeExample1 {  
  
    // Method with same name and same data type of  
    // integer parameters  
    public int function_name_add(int a, int b)  
    {  
        // Returns sum of integer numbers  
        return (a + b);  
    }  
  
    public static void main(String[] args)  
    {  
        RunTimeExample2 obj = new RunTimeExample2( );  
        // Calling method by passing input as in arguments  
        System.out.println(obj.function_name_add(2, 4));  
    }  
}
```



## **PROGRAMS –**

### **QUESTION 1 –**



### **CODE –**

#### **Peri\_Square.java**

```
package Practs2;
import java.util.*;
import java.io.*;

public class Peri_Square {
    int s, peri;
    Scanner sc = new Scanner(System.in);
    public void calculate_peri_square()
    {
        System.out.println("* CALCULATING PERIMETER OF SQUARE *\n");
        System.out.print("Enter side of the square : ");
        s = sc.nextInt();
        peri = 4*s;
    }
    public void display_peri_square()
    {
        System.out.println("Perimeter of the square with side " + s + " = " + peri);
    }
}
```

#### **Perimeter.java**

```
package Practs2;
public class Perimeter extends Peri_Square {
    public static void main(String[] args) {
        System.out.println("* SINGLE INHERITANCE *");
        Perimeter p = new Perimeter();
        p.calculate_peri_square();
        p.display_peri_square();
    }
}
```

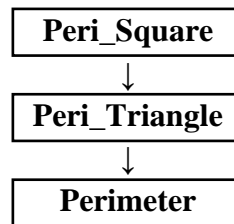
## OUTPUT –

```
run:
* SINGLE INHERITANCE *
* CALCULATING PERIMETER OF SQUARE *

Enter side of the square : 9
Perimeter of the square with side '9' = 36
BUILD SUCCESSFUL (total time: 3 seconds)
```

**Figure 2:** Demonstration of Single Inheritance by calculating the perimeter of a square

## QUESTION 2 –



## CODE –

### **Peri\_Square.java**

```
package Practs2;
import java.util.*;
import java.io.*;

public class Peri_Square {
    int s, peri;
    Scanner sc = new Scanner(System.in);

    public void calculate_peri_square()
    {
        System.out.println("*****");
        System.out.println("* CALCULATING PERIMETER OF SQUARE *");
        System.out.println("*****\n");
        System.out.print("Enter side of the square : ");
        s = sc.nextInt();
        peri = 4*s;
        System.out.println("Perimeter of the square with side '" + s + "' = " + peri);
    }
}
```

**Peri\_Triangle.java**

```
package Practs2;
import java.util.*;
import java.io.*;

public class Peri_Triangle extends Peri_Square {
    Scanner sc = new Scanner(System.in);
    int side1, side2, side3, peri;

    public void calculate_peri_triangle()
    {
        System.out.println("\n*****");
        System.out.println("* CALCULATING PERIMETER OF TRIANGLE *");
        System.out.println("*****\n");
        System.out.print("Enter first side of the triangle : ");
        side1 = sc.nextInt();
        System.out.print("Enter second side of the triangle : ");
        side2 = sc.nextInt();
        System.out.print("Enter third side of the triangle : ");
        side3 = sc.nextInt();
        peri = side1 + side2 + side3;
        System.out.println("Perimeter of the triangle with sides '" + side1 + "', '" + side2
+ "' and '" + side3 + "' = " + peri + "\n");
    }
}
```

**Perimeter.java**

```
package Practs2;
public class Perimeter extends Peri_Triangle {

    public static void main(String[] args) {
        System.out.println("*****");
        System.out.println("* MULTILEVEL INHERITANCE *");
        System.out.println("*****\n");
        Perimeter p = new Perimeter();
        p.calculate_peri_square();
        p.calculate_peri_triangle();
    }
}
```

## OUTPUT –

```
run:
*****
* MULTILEVEL INHERITANCE *
*****

*****
* CALCULATING PERIMETER OF SQUARE *
*****

Enter side of the square : 9
Perimeter of the square with side '9' = 36

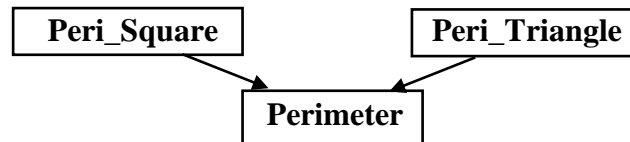
*****
* CALCULATING PERIMETER OF TRIANGLE *
*****

Enter first side of the triangle : 4
Enter second side of the triangle : 5
Enter third side of the triangle : 12
Perimeter of the triangle with sides '4', '5' and '12' = 21

BUILD SUCCESSFUL (total time: 15 seconds)
```

**Figure 3:** Demonstration of Multilevel Inheritance by calculating the perimeter of a square and a triangle

## QUESTION 3 –



## CODE –

### Peri\_Square1.java

```
package Practs2;

public interface Peri_Square1 {
    public void calc_peri_square();
}
```

### Peri\_Triangle1.java

```
package Practs2;

public interface Peri_Triangle1 {
    public void calc_peri_triangle();
}
```

**Perimeter.java**

```
package Practs2;
import java.util.*;
import java.io.*;

public class Perimeter implements Peri_Square1, Peri_Triangle1 {
    int s, side1, side2, side3, peri;
    Scanner sc = new Scanner(System.in);

    public void calc_peri_square()
    {
        System.out.println("*****");
        System.out.println("* CALCULATING PERIMETER OF SQUARE *");
        System.out.println("*****\n");
        System.out.print("Enter side of the square : ");
        s = sc.nextInt();
        peri = 4*s;
        System.out.println("Perimeter of the square with side '" + s + "' = " + peri);
    }

    public void calc_peri_triangle()
    {
        System.out.println("\n*****");
        System.out.println("* CALCULATING PERIMETER OF TRIANGLE *");
        System.out.println("*****\n");
        System.out.print("Enter first side of the triangle : ");
        side1 = sc.nextInt();
        System.out.print("Enter second side of the triangle : ");
        side2 = sc.nextInt();
        System.out.print("Enter third side of the triangle : ");
        side3 = sc.nextInt();
        peri = side1 + side2 + side3;
        System.out.println("Perimeter of the triangle with sides '" + side1 + "', '" + side2
+ "' and '" + side3 + "' = " + peri + "\n");
    }

    public static void main(String[] args) {
        System.out.println("*****");
        System.out.println("* MULTIPLE INHERITANCE *");
        System.out.println("*****\n");
        Perimeter p = new Perimeter();
        p.calc_peri_square();
        p.calc_peri_triangle();
    }
}
```

## OUTPUT –

```
run:
*****
* MULTIPLE INHERITANCE *
*****

*****
* CALCULATING PERIMETER OF SQUARE *
*****

Enter side of the square : 4
Perimeter of the square with side '4' = 16

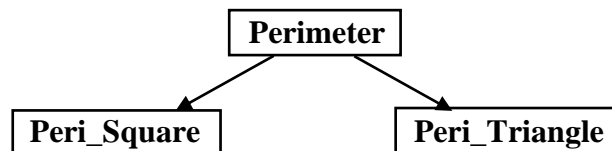
*****
* CALCULATING PERIMETER OF TRIANGLE *
*****

Enter first side of the triangle : 8
Enter second side of the triangle : 5
Enter third side of the triangle : 2
Perimeter of the triangle with sides '8', '5' and '2' = 15

BUILD SUCCESSFUL (total time: 6 seconds)
|
```

**Figure 4:** Demonstration of Multiple Inheritance by calculating the perimeter of a square and a triangle

## QUESTION 4 –



## CODE –

### Perimeter.java

```
package Practs2;
import java.util.*;
import java.io.*;

public class Perimeter1 {
    public void peri()
    {
        System.out.println("*****
*****");
        System.out.println("* HIERARCHICAL INHERITANCE FOR
CALCULATING PERIMETER*");
    }
}
```

```

        System.out.println("*****\n");
    }
}

```

#### **Peri\_Square.java**

```

package Practs2;
import java.util.*;
import java.io.*;

public class Peri_Square2 extends Perimeter1 {

    public static void main(String[] args) {
        Peri_Square2 obj = new Peri_Square2();
        obj.peri();

        Scanner sc = new Scanner(System.in);
        int s, peri;
        System.out.println("*****");
        System.out.println("* CALCULATING PERIMETER OF SQUARE *");
        System.out.println("*****\n");
        System.out.print("Enter side of the square : ");
        s = sc.nextInt();
        peri = 4*s;
        System.out.println("Perimeter of the square with side " + s + " = " + peri);
    }
}

```

#### **Peri\_Triangle.java**

```

package Practs2;
import java.util.*;
import java.io.*;

public class Peri_Triangle2 extends Perimeter1 {

    public static void main(String[] args) {
        Peri_Triangle2 obj = new Peri_Triangle2();
        obj.peri();
        System.out.println("\n*****");
        System.out.println("* CALCULATING PERIMETER OF TRIANGLE *");
        System.out.println("*****\n");
        Scanner sc = new Scanner(System.in);
        int side1, side2, side3, peri;
        System.out.print("Enter first side of the triangle : ");
    }
}

```

```

side1 = sc.nextInt();
System.out.print("Enter second side of the triangle : ");
side2 = sc.nextInt();
System.out.print("Enter third side of the triangle : ");
side3 = sc.nextInt();
peri = side1 + side2 + side3;
System.out.println("Perimeter of the triangle with sides '" + side1 + "', '" + side2
+ "' and '" + side3 + "' = " + peri + "\n");
    }
}

```

**OUTPUT –**

```

run:
*****
* HIERARCHICAL INHERITANCE FOR CALCULATING PERIMETER *
*****

*****
* CALCULATING PERIMETER OF SQUARE *
*****

Enter side of the square : 5
Perimeter of the square with side '5' = 20
BUILD SUCCESSFUL (total time: 2 seconds)

```

```

run:
*****
* HIERARCHICAL INHERITANCE FOR CALCULATING PERIMETER *
*****

*****
* CALCULATING PERIMETER OF TRIANGLE *
*****

Enter first side of the triangle : 3
Enter second side of the triangle : 6
Enter third side of the triangle : 9
Perimeter of the triangle with sides '3', '6' and '9' = 18

BUILD SUCCESSFUL (total time: 13 seconds)

```

**Figure 5:** Demonstration of Hierarchical Inheritance by calculating the perimeter of a square and a triangle



### QUESTION 5 –

Write a java program to create class 'Area' and method named as 'calarea()' to find area of a square and rectangle using method overloading.

### CODE –

```
package Practs2;
import java.io.*;
import java.util.*;

public class Area {

    public int calarea(int side)
    {
        int area = side * side;
        return area;
    }

    public int calarea(int length, int breadth)
    {
        int area = length * breadth;
        return area;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("*****");
        System.out.println("* METHOD OVERLOADING *");
        System.out.println("*****\n");
        Area obj = new Area();

        System.out.println("*****");
        System.out.println("* CALCULATING AREA OF SQUARE *");
        System.out.println("*****\n");
        System.out.print("Enter side of square: ");
        int side = sc.nextInt();
        System.out.println("Area of square = " + obj.calarea(side));

        System.out.println("\n*****");
        System.out.println("* CALCULATING AREA OF RECTANGLE *");
        System.out.println("*****\n");
        System.out.print("Enter side of length: ");
        int length = sc.nextInt();
        System.out.print("Enter side of breadth: ");
        int breadth = sc.nextInt();
        System.out.println("Area of rectangle = " + obj.calarea(length, breadth));

    }
}
```

## OUTPUT –

```
run:
*****
* METHOD OVERLOADING *
*****

*****
* CALCULATING AREA OF SQUARE *
*****

Enter side of square: 9
Area of square = 81

*****
* CALCULATING AREA OF RECTANGLE *
*****

Enter side of length: 5
Enter side of breadth: 6
Area of rectangle = 30
BUILD SUCCESSFUL (total time: 9 seconds)
|
```

**Figure 6:** Demonstration of Method Overloading by calculating the area of a square and a rectangle

## QUESTION 6 –

Write a java program to create class ‘Calculator’ and method named as ‘add()’ – for addition of two numbers and create another class ‘Addition’ that inherits class ‘Calculator’ and method named as ‘add()’ – for addition of three numbers using overriding.

## CODE –

### Calculator.java

```
package Practs2;
import java.io.*;
import java.util.*;

public class Calculator {
    public void add()
    {
        int num1, num2, sum;
        Scanner sc = new Scanner(System.in);
        System.out.println("*****");
        System.out.println("* CALCULATING SUM OF TWO NUMBERS *");
        System.out.println("*****\n");
        System.out.print("Enter 1st number: ");
        num1 = sc.nextInt();
        System.out.print("Enter 2nd number: ");
```

```

        num2 = sc.nextInt();
        sum = num1 + num2;
        System.out.println("Sum of 2 numbers (" + num1 + " and " + num2 + ") = " +
sum + "\n");
    }
}

```

### **Addition.java**

```

package Practs2;
import java.io.*;
import java.util.*;

public class Addition extends Calculator {

    public void add()
    {
        int num1, num2, num3, sum;
        Scanner sc = new Scanner(System.in);
        System.out.println("*****");
        System.out.println("* CALCULATING SUM OF THREE NUMBERS *");
        System.out.println("*****\n");
        System.out.print("Enter 1st number: ");
        num1 = sc.nextInt();
        System.out.print("Enter 2nd number: ");
        num2 = sc.nextInt();
        System.out.print("Enter 3rd number: ");
        num3 = sc.nextInt();
        sum = num1 + num2 + num3;
        System.out.println("\nSum of 3 numbers (" + num1 + ", " + num2 + " and " +
num3 + ") = " + sum);
    }

    public static void main(String[] args) {
        Calculator obj = new Calculator();
        obj.add();
        obj = new Addition();
        obj.add();
    }
}

```

## OUTPUT –

```
run:
*****
* CALCULATING SUM OF TWO NUMBERS *
*****

Enter 1st number: 2
Enter 2nd number: 3
Sum of 2 numbers (2 and 3) = 5

*****
* CALCULATING SUM OF THREE NUMBERS *
*****

Enter 1st number: 7
Enter 2nd number: 5
Enter 3rd number: 3

Sum of 3 numbers (7, 5 and 3) = 15
BUILD SUCCESSFUL (total time: 13 seconds)
```

**Figure 7:** Demonstration of Method Overriding by calculating the sum of two numbers and three numbers

## **RESULTS:**

Using the Java programming language, the concept of inheritance and polymorphism were explored and demonstrated in order to calculate the perimeter of a square using single inheritance, to calculate the perimeter of a square and a triangle using multilevel inheritance, multiple inheritance and hierarchical inheritance, to calculate the area of a square and a rectangle using method overloading and to calculate the sum of two numbers and 3 numbers using method overriding.

## **CONCLUSION:**

The concept of inheritance and polymorphism of Java programming language were explored and demonstrated.

## **REFERENCES:**

1. G. (2023, November 10). Inheritance in Java. GeeksforGeeks. <https://www.geeksforgeeks.org/inheritance-in-java/>
2. Inheritance in Java - Javatpoint. (n.d.). www.javatpoint.com. <https://www.javatpoint.com/inheritance-in-java>
3. Types of Inheritance in Java - Javatpoint. (n.d.). www.javatpoint.com. <https://www.javatpoint.com/types-of-inheritance-in-java>
4. Ramuglia, G. (2024, February 20). Java Polymorphism: Class and Object Manipulation Guide. Linux Dedicated Server Blog. <https://ioflood.com/blog/polymorphism-java/#:~:text=The%20Role%20of%20Polymorphism%20in,code%20more%20flexible%20and%20dynamic.>

5. G. (2023, November 1). Polymorphism in Java. GeeksforGeeks.  
<https://www.geeksforgeeks.org/polymorphism-in-java/>
  6. S. (2023, February 24). What Is Polymorphism in Java and How to Implement It?  
Simplilearn.com. <https://www.simplilearn.com/tutorials/java-tutorial/java-polymorphism>
-

### Practical 3

## JAVA: Arrays, Loops and Conditional Statements

### AIM:

To explore and demonstrate the concept of arrays, loops and conditional statements in Java programming language.

### INTRODUCTION:

Java is a versatile programming language released in 1995 that runs on various platforms without needing recompilation. Its object-oriented design and platform independence make it popular for diverse applications, from web development to mobile apps. Known for its "write once, run anywhere" capability, Java simplifies development and promotes code reusability across systems.

### ARRAYS

Arrays in Java are objects that are dynamically created and can be assigned to variables of type Object. They contain a number of variables accessed by non-negative integer index values, with all components having the same type, known as the component type of the array. They are data structures used to store multiple values of similar data types in a contiguous memory location. A number of arithmetic and dynamic programming methods rely on arrays.

**Advantage** – Along with the dynamic creation of memory, hence reducing storage requirements, arrays allow random access to elements using indices and facilitate the storage and manipulation of large datasets efficiently.

**Disadvantage** – Arrays in Java programming language have two limitations: they are of fixed size (once declared, their size cannot be changed) and can only store elements of a single data type (not heterogeneous data).

### **Types of Arrays –**

Type	Description	Syntax	Example
One – dimensional array	It is a group of elements having the same datatype which are stored in a linear arrangement under a single variable name. It requires only one indices specification in order to access a particular element of the array.	<code>data_type array_name [] = new data_type [array_size];</code>	<code>int arr[] = new int [3];</code>
Two – dimensional array	It is an array of arrays, allowing data to be stored in a tabular format with rows and columns. Each element is associated with both a row number and a column number.	<code>data_type array_name [][] = new data_type [row_size][column_size];</code>	<code>int arr[][] = new int [3][3];</code>

## LOOPS

Loops enable the repeated execution of a sequence of instructions until a predetermined condition is met. Following are the parts of a loop –

1. **Initialization Expression** – initializes the loop variables in the beginning of the loop.
2. **Test Expression** – decides whether the loop will be executed (if the test expression is true) or not (if the test expression is false).
3. **Update Expressions** – update (increment / decrement) the values of the loop variables after every iteration of the loop.
4. **Body of the loop** – contains a set of statements to be executed repeatedly.

### Types of Loops

#### 1. Entry – Controlled Loops –

In this type of loop, the test expression is checked before entering the body of the loop.

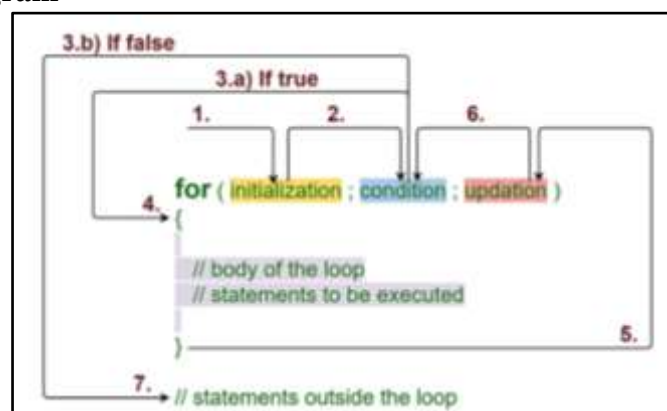
##### (a) for loop –

A **for loop** is a repetition control structure that allows us to write a loop that is executed a specific / pre – determined number of times. The loop enables us to perform ‘n’ number of steps sequentially in one line.

**Syntax** – for (initialization\_expression ; test\_expression ; update\_expression)

```
{  
    // code to be executed  
}
```

**Diagram –**



**Figure 1: Working of for loop**

##### (b) while loop –

A **while loop** repeats a statement or group of statements until a given condition is true. It tests the condition before executing the loop body. The loop consists of the test expression whereas the initialization expression and update expressions are addressed elsewhere.

**Syntax** – initialization\_expression;

while(test\_expression)

```
{  
    // code to be executed  
    update_expression;  
}
```

Diagram –

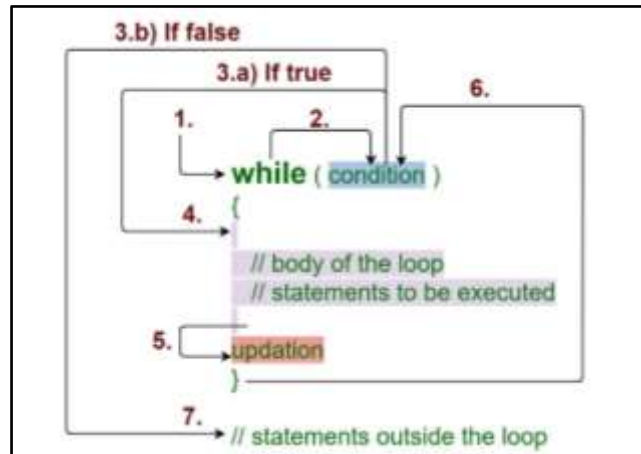


Figure 2: Working of while loop

## 2. Exit – Controlled Loops –

In this type of loop, the test condition is checked at the end of the loop body. Therefore, the loop body will execute a set of statements at least once, irrespective of whether the test expression is true or false.

**do – while loop** is a type of exit – controlled loop where the test expression is tested at the end of the loop body and the loop is terminated on the basis of the test conditions after executing the loop body at least once.

**Syntax** – initialization\_expression;

do

{

// code to be executed

update\_expression;

} while(test\_expression);

Diagram –

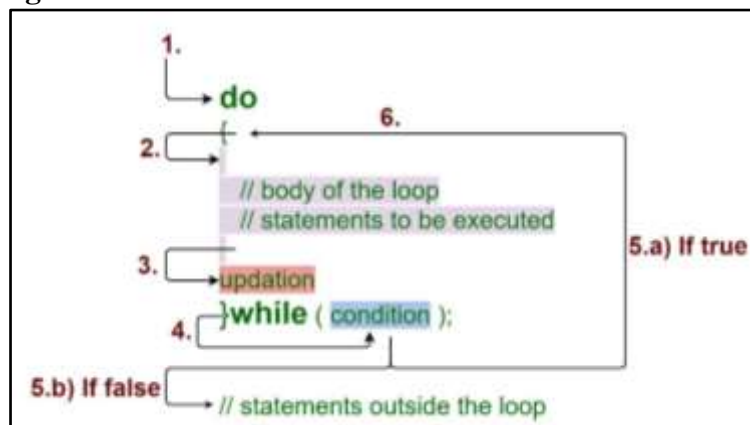


Figure 3: Working of do – while loop



## CONDITIONAL STATEMENTS

Conditional statements, also referred to as decision control structures, are utilized for decision – making purposes in Java programs. These conditional statements are employed to assess and evaluate one or more conditions / criteria and determine whether or not to execute a series of statements or a specific block of code. Such decision – making statements in programming languages decide the direction of the flow of program execution.

### Types of Conditional Statements

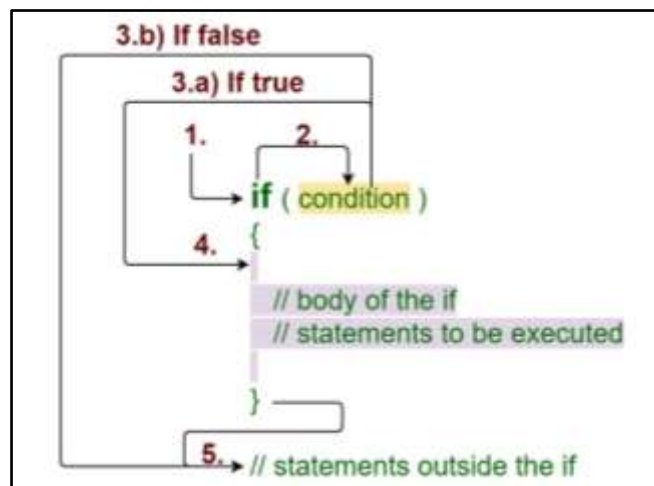
#### 1. if Statement –

The **if** conditional statement is used to decide whether or not a specific statement or block of statements will be executed if a particular condition is true.

**Syntax** – if (condition)

```
{  
    // code to be executed if the condition returns true  
}
```

**Diagram** –



**Figure 4:** Working of **if** conditional statement

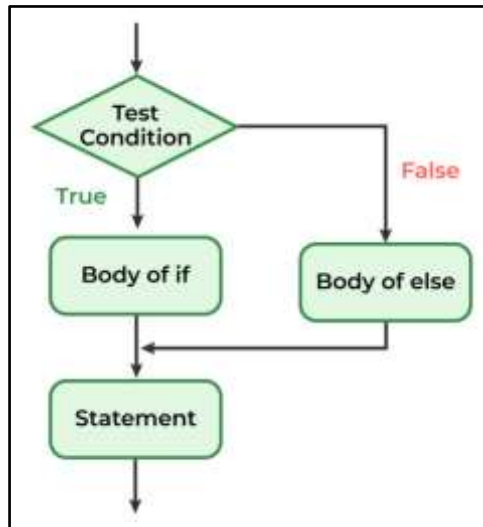
#### 2. if – else Statement –

The **if - else conditional statement** is a two – way branching statement. It consists of two blocks of statements each enclosed inside **if block** and **else block** respectively. If the condition inside **if statement** is true, statements inside **if block** are executed, otherwise statements inside **else block** are executed.

**Syntax** – if (condition)

```
{  
    // code to be executed if the condition returns true  
}  
else  
{  
    // code to be executed if the condition returns false  
}
```

**Diagram –**



**Figure 5:** Working of **if – else** conditional statement

**3. if – else – if Ladder –**

The **if – else – if Ladder** is used when more than one condition is to be checked. A block of statement is enclosed inside **if**, **else if** and **else** part. Conditions are checked in each **if** and **else if** part. If the condition is true, the statements inside that block are executed. If none of the conditions are true, the statements inside else block are executed. Such a statement must have only one **if block** but can have as many **else if block** as required.

**Syntax –** if (condition1)

```
{  
    // code to be executed if condition1 returns true  
}  
else if (condition2)  
{  
    // code to be executed if condition1 returns false  
    // and condition2 returns true  
}  
  
else if (conditionN)  
{  
    // code to be executed if previous conditions return false  
    // and conditionN returns true  
}  
else  
{  
    // code to be executed if all the previous conditions return false  
}
```

Diagram –

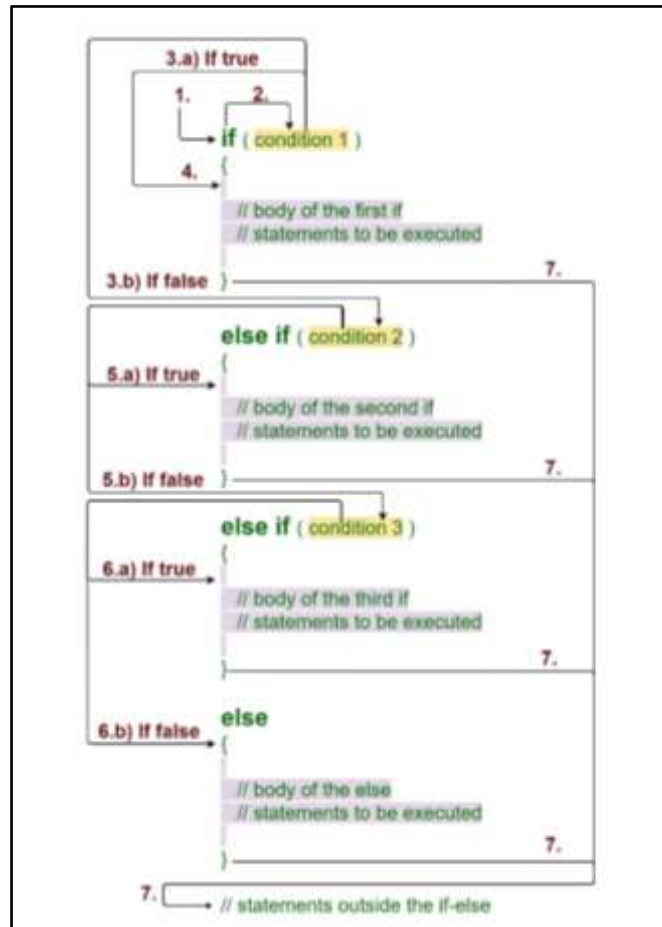


Figure 6: Working of if – else – if Ladder

#### 4. Nested if Statement –

When an **if statement** is kept inside another **if statement**, it is called nested if statement. **Nested if statements** are used if there is a sub condition to be tested.

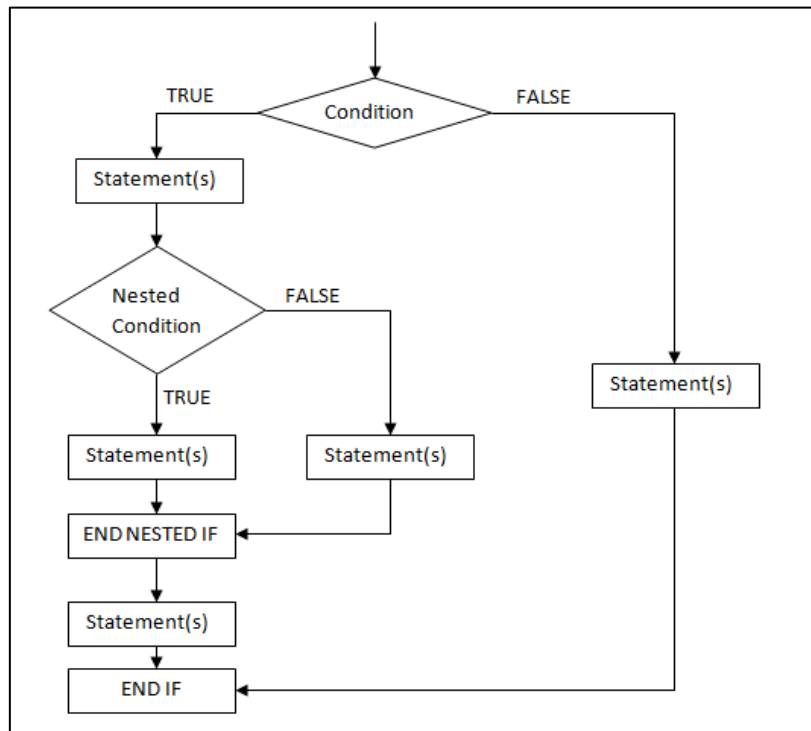
**Syntax** – if (condition1)

```

{
    // Nested if statement to be executed if condition1 returns true
    if (sub_condition1)
    {
        // code to be executed if sub_condition1 returns true
    }
}
else if (condition2)
{
    // code to be executed if condition1 returns false
    // and condition2 returns true
    // Nested if statement to be executed if condition2 returns true
    if (sub_condition2)
    {
        // code to be executed if sub_condition2 returns true
    }
}

```

**Diagram –**



**Figure 7: Working of Nested if conditional statement**

## **PROGRAMS:**

### **Question 1 –**

Write a java program to calculate the sum of all elements of an array.

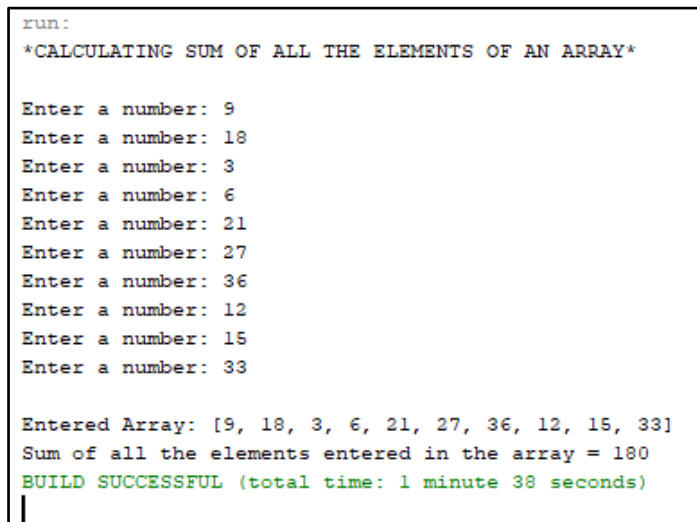
### **CODE –**

```
package Practs3;
import java.util.*;
import java.io.*;

public class SumOfArrayElements {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i, num, sum = 0;
        int array1[] = new int[10];
        System.out.println("*CALCULATING SUM OF ALL THE ELEMENTS OF
AN ARRAY*\n");
        for(i=0;i<10;i++)
        {
            System.out.print("Enter a number: ");
            array1[i] = sc.nextInt();
            sum += array1[i];
        }
        System.out.print("\nEntered Array: [");
        for(i=0;i<9;i++)
        {
            System.out.print(array1[i] + ", ");
        }
        System.out.println(array1[9] + "]");
        System.out.println("Sum of all the elements entered in the array = " + sum);
    }
}
```

### **OUTPUT –**

A screenshot of a Java program execution. The output shows a title line, followed by ten prompts to enter numbers. The entered numbers are 9, 18, 3, 6, 21, 27, 36, 12, 15, and 33. Then, the entered array is displayed as [9, 18, 3, 6, 21, 27, 36, 12, 15, 33], and the sum of all elements is calculated as 180. The execution ends with a success message and a build time of 1 minute 38 seconds.

```
run:
*CALCULATING SUM OF ALL THE ELEMENTS OF AN ARRAY*

Enter a number: 9
Enter a number: 18
Enter a number: 3
Enter a number: 6
Enter a number: 21
Enter a number: 27
Enter a number: 36
Enter a number: 12
Enter a number: 15
Enter a number: 33

Entered Array: [9, 18, 3, 6, 21, 27, 36, 12, 15, 33]
Sum of all the elements entered in the array = 180
BUILD SUCCESSFUL (total time: 1 minute 38 seconds)
|
```

**Figure 8:** Calculating and displaying the sum of all the elements of an array

### Question 2 –

Write a program in java that takes a number as input and prints its multiplication table upto 10.

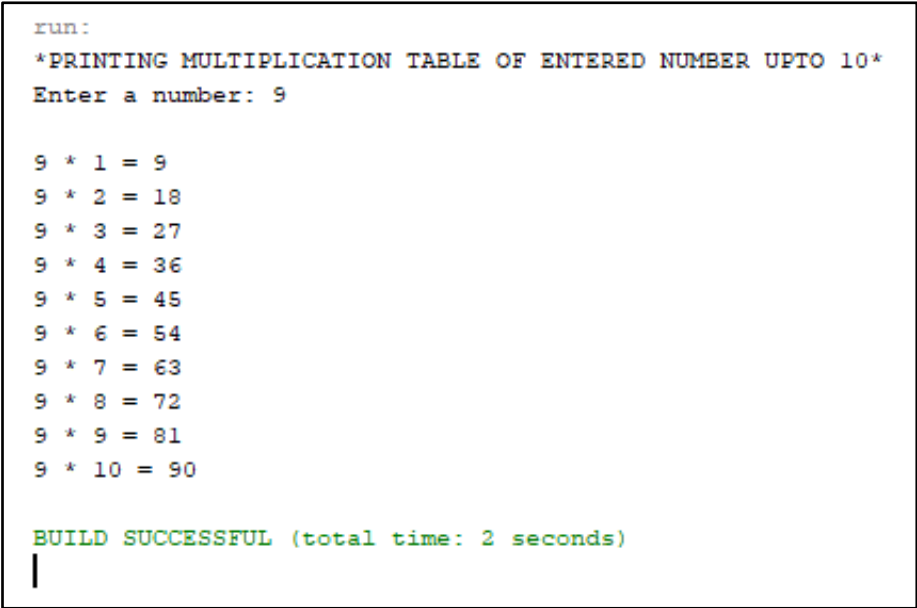
### CODE –

```
package Practs3;
import java.util.*;
import java.io.*;

public class MultiplicationTable {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num, i;
        System.out.println("*PRINTING MULTIPLICATION TABLE OF ENTERED
NUMBER UPTO 10*");
        System.out.print("Enter a number: ");
        num = sc.nextInt();
        System.out.println("");
        for(i=1;i<=10;i++)
        {
            System.out.println(num + " * " + i + " = " + (num*i));
        }
        System.out.println("");
    }
}
```

### OUTPUT –



```
run:
*PRINTING MULTIPLICATION TABLE OF ENTERED NUMBER UPTO 10*
Enter a number: 9

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

BUILD SUCCESSFUL (total time: 2 seconds)
|
```

**Figure 9:** Calculating and displaying the multiplication table of a user – entered number upto

**Question 3 –**

Write a java program to accept angles of a triangle and display whether it is an equilateral, isosceles or scalene triangle.

**CODE –**

```
package Practs3;
import java.util.*;
import java.io.*;

public class Triangle {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        boolean flag = true;
        System.out.println("*DETERMINING THE TYPE OF TRIANGLE BASED ON
ITS ANGLES*");
        int angle1, angle2, angle3;

        while(flag)
        {
            System.out.print("Enter value of angle1 : ");
            angle1 = sc.nextInt();
            System.out.print("Enter value of angle2 : ");
            angle2 = sc.nextInt();
            System.out.print("Enter value of angle3 : ");
            angle3 = sc.nextInt();

            if(angle1 + angle2 + angle3 == 180)
            {
                flag = false;
                if(angle1 == angle2 && angle2 == angle3 && angle3 == angle1)
                {
                    System.out.println("\nEntered angles represent an EQUILATERAL
TRIANGLE");
                }
                else if(angle1 == angle2 || angle2 == angle3 || angle3 == angle1)
                {
                    System.out.println("\nEntered angles represent an ISOSCELES
TRIANGLE");
                }
                else if(angle1 != angle2 && angle2 != angle3 && angle3 != angle1)
                {
                    System.out.println("\nEntered angles represent an SCALENE
TRIANGLE");
                }
            }
        }
    }
}
```

```

    }
}
else
{
    System.out.println("\nEntered angles do not represent a triangle");
    System.out.println("Enter the values of the angles again\n");
}
}
}
}
}

```

**OUTPUT –**

```

run:
*DETERMINING THE TYPE OF TRIANGLE BASED ON ITS ANGLES*
Enter value of angle1 : 95
Enter value of angle2 : 35
Enter value of angle3 : 23

Entered angles do not represent a triangle
Enter the values of the angles again

Enter value of angle1 : 60
Enter value of angle2 : 60
Enter value of angle3 : 60

Entered angles represent an EQUILATERAL TRIANGLE
BUILD SUCCESSFUL (total time: 28 seconds)
|

```

**Figure 10:** Type of triangle determined based on its angles: Equilateral triangle

```

run:
*DETERMINING THE TYPE OF TRIANGLE BASED ON ITS ANGLES*
Enter value of angle1 : 10
Enter value of angle2 : 20
Enter value of angle3 : 30

Entered angles do not represent a triangle
Enter the values of the angles again

Enter value of angle1 : 45
Enter value of angle2 : 45
Enter value of angle3 : 90

Entered angles represent an ISOSCELES TRIANGLE
BUILD SUCCESSFUL (total time: 26 seconds)

```

**Figure 11:** Type of triangle determined based on its angles: Isosceles triangle



```
run:
*DETERMINING THE TYPE OF TRIANGLE BASED ON ITS ANGLES*
Enter value of angle1 : 15
Enter value of angle2 : 25
Enter value of angle3 : 90

Entered angles do not represent a triangle
Enter the values of the angles again

Enter value of angle1 : 98
Enter value of angle2 : 55
Enter value of angle3 : 15

Entered angles do not represent a triangle
Enter the values of the angles again

Enter value of angle1 : 53
Enter value of angle2 : 37
Enter value of angle3 : 90

Entered angles represent an SCALENE TRIANGLE
BUILD SUCCESSFUL (total time: 52 seconds)
```

**Figure 12:** Type of triangle determined based on its angles: Scalene triangle

**Question 4 –**

Write a java program to count the number of occurrences of given number in an array of integers.

**CODE –**

```
package Practs3;
import java.util.*;
import java.io.*;

public class NumberOfOccurrences {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int array1[] = new int[10];
        int i, num, count = 0;
        System.out.println("*COUNTING THE NUMBER OF OCCURRENCES OF A
        GIVEN NUMBER IN A USER-ENTERED ARRAY*\n");
        for(i=0;i<10;i++)
        {
            System.out.print("Enter a number: ");
            array1[i] = sc.nextInt();
        }

        System.out.print("\nEnter a number to count its occurrences in the array: ");
        num = sc.nextInt();
        for(i=0;i<10;i++)
        {
            if(array1[i] == num)
            {
                count++;
            }
        }

        System.out.println("\n*****
        *****");
        System.out.print("USER-ENTERED ARRAY: [");
        for(i=0;i<9;i++)
        {
            System.out.print("'" + array1[i] + ", ");
        }
        System.out.println(array1[9] + "']");
        System.out.println("Number of occurrences of the number '" + num + "' in the
        array = " + count);
    }
}
```

```

System.out.println("*****
*****\n");
    }
}

```

## OUTPUT –

```

run:
*COUNTING THE NUMBER OF OCCURRENCES OF A GIVEN NUMBER IN A USER-ENTERED ARRAY*

Enter a number: 3
Enter a number: 6
Enter a number: 9
Enter a number: 5
Enter a number: 9
Enter a number: 2
Enter a number: 8
Enter a number: 9
Enter a number: 9
Enter a number: 7

Enter a number to count its occurrences in the array: 9

*****
USER-ENTERED ARRAY: [3, 6, 9, 5, 9, 2, 8, 9, 9, 7]
Number of occurrences of the number '9' in the array = 4
*****

BUILD SUCCESSFUL (total time: 24 seconds)

```

**Figure 13:** Counting and displaying the number of occurrences of a given number in a user – entered array

**Question 5 –**

Write a java program to accept 3x3 matrix and display the Transpose of a given matrix.

**CODE –**

```
package Practs3;
import java.util.*;
import java.io.*;

public class Transpose {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("*DISPLAYING TRANSPOSE OF A USER-ENTERED
MATRIX*\n");
        int array1[][] = new int[3][3];
        int i, j;

        for(i=0;i<3;i++)
        {
            for(j=0;j<3;j++)
            {
                System.out.print("Enter value for position (" + i + ", " + j + "): ");
                array1[i][j] = sc.nextInt();
            }
        }

        System.out.println("\nORIGINAL MATRIX: ");
        for(i=0;i<3;i++)
        {
            for(j=0;j<3;j++)
            {
                System.out.print(array1[i][j] + "\t");
            }
            System.out.println("");
        }

        System.out.println("\nTRANSPOSE MATRIX: ");
        for(i=0;i<3;i++)
        {
            for(j=0;j<3;j++)
            {
                System.out.print(array1[j][i] + "\t");
            }
            System.out.println("");
        }
    }
}
```

```

    }
    System.out.println("");
}
}

```

## OUTPUT –

```

run:
*DISPLAYING TRANSPOSE OF A USER-ENTERED MATRIX*

Enter value for position (0, 0): 9
Enter value for position (0, 1): 6
Enter value for position (0, 2): 3
Enter value for position (1, 0): 8
Enter value for position (1, 1): 5
Enter value for position (1, 2): 2
Enter value for position (2, 0): 7
Enter value for position (2, 1): 4
Enter value for position (2, 2): 1

ORIGINAL MATRIX:
9      6      3
8      5      2
7      4      1

TRANSPOSE MATRIX:
9      8      7
6      5      4
3      2      1

BUILD SUCCESSFUL (total time: 1 minute 7 seconds)
|

```

**Figure 14:** Displaying the transpose of a user – entered matrix

## RESULTS:

Using the Java programming language, the concept of arrays, loops and conditional statements was explored and demonstrated in order to calculate the sum of all elements of a user – entered array, to calculate and display the multiplication table of a user – entered number upto 10, to determine the type of triangle (equilateral, isosceles or scalene triangle) based on its angles, to count and display the total number of occurrences of a user – entered number in an array and to display the transpose of a user – entered matrix.

## CONCLUSION:

The concept of arrays, loops and conditional statements of Java programming language were demonstrated.

## **REFERENCES:**

1. *Chapter 10. Arrays.* (n.d.). <https://docs.oracle.com/javase/specs/jls/se7/html/jls-10.html>
  2. *Arrays (The Java™ Tutorials > Learning the Java Language > Language Basics).* (n.d.). <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>
  3. G. (2023, December 13). *Arrays in Java.* GeeksforGeeks. <https://www.geeksforgeeks.org/arrays-in-java/?ref=lbp>
  4. G. (2023, June 12). *Loops in Java.* GeeksforGeeks. <https://www.geeksforgeeks.org/loops-in-java/>
  5. *Java If . . . Else.* (n.d.). [https://www.w3schools.com/java/java\\_conditions.asp](https://www.w3schools.com/java/java_conditions.asp)
  6. G. (2023, February 18). *Decision Making in Java (if, if-else, switch, break, continue, jump).* GeeksforGeeks. <https://www.geeksforgeeks.org/decision-making-javaif-else-switch-break-continue-jump/>
-

## **Practical 4**

### **JAVA: Abstract Window Toolkit (AWT)**

#### **AIM:**

To explore and demonstrate the concept of Abstract Window Toolkit (AWT) by creating GUI – based programs in Java programming language.

#### **INTRODUCTION:**

Java is a versatile programming language released in 1995 that runs on various platforms without needing recompilation. Its object-oriented design and platform independence make it popular for diverse applications, from web development to mobile apps. Known for its "write once, run anywhere" capability, Java simplifies development and promotes code reusability across systems.

#### **Java Abstract Window Toolkit (AWT)**

The Abstract Window Toolkit (AWT) in Java is an API that enables developers to create Graphical User Interfaces (GUIs) or Windows-based applications. The AWT package, which is part of the Java Foundation Classes (JFC), is used to create user interfaces for a variety of Java applications. AWT provides a variety of GUI components, including buttons, labels, text fields, checkboxes, and containers such as frames and panels. These components enable developers to create dynamic and visually appealing interfaces for Java applications.

#### **Advantages of Java AWT –**

1. **Platform Independence** – Despite being platform-dependent in terms of appearance, AWT ensures consistency across different environments by utilizing native platform subroutines to create components.
2. **Integration** – AWT can be easily integrated with other Java libraries and APIs, extending its functionality beyond basic GUI components.
3. **Graphics and Drawing** – AWT includes classes for graphics and drawing operations, enabling developers to create custom graphics, charts, and images, which is beneficial for data visualizations and custom GUI elements.

#### **Disadvantages of Java AWT –**

1. **Heavyweight Nature** – AWT is considered heavyweight due to its reliance on the system's host operating system, which can impact performance and memory consumption.
2. **Missing Components** – AWT lacks important components like tables and trees that are commonly used in desktop applications, limiting its functionality in certain scenarios.
3. **Lack of Modern features** – AWT, being one of the earliest GUI libraries in Java, lacks some modern features found in later frameworks like JavaFX, such as hardware acceleration, which can impact graphical performance.

## Key features of Java AWT

### 1. Components –

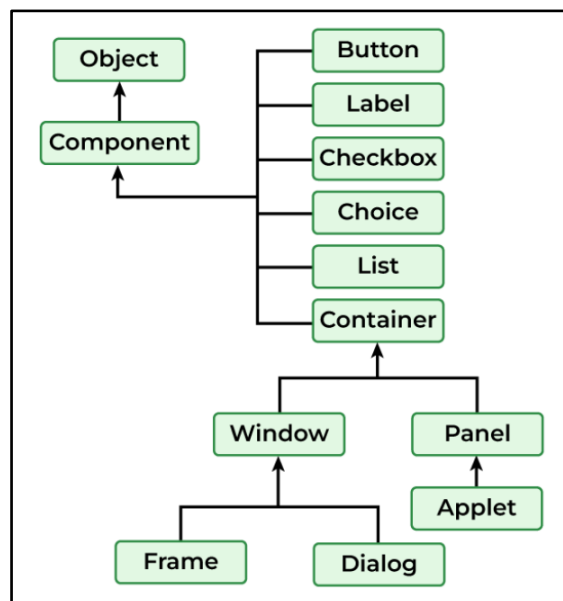
The Component class in Java AWT is an abstract base class for non – menu user – interface controls. It represents an object with a graphical representation and serves as the foundation for various GUI components like buttons, labels, text fields, and checkboxes within the AWT framework.

### 2. Containers –

AWT offers container classes like Frame, Panel, and Window for grouping and organizing components within an application.

Types of Containers include –

- (a) **Window** – a top – level container that represents a graphical window or a dialog box.
- (b) **Panel** – a lightweight container used for grouping other components together within a window or a frame.
- (c) **Frame** – a container that contains the title bar and border and can have menu bars.
- (d) **Dialog** – a temporary window an application creates to retrieve user input.



**Figure 1:** Hierarchy of AWT Components

### 3. Layout Managers –

AWT includes layout managers like FlowLayout, BorderLayout, and GridLayout to help organize and arrange components within containers effectively.

### 4. Event Handling –

AWT supports event handling mechanisms for managing user interactions such as mouse clicks, key presses, and window events. Handlers are automatically called when an event takes place. Event handlers called ‘Listeners’ are created with objects whose events need to be captured and are implemented as Interfaces in Java.



Event Class	Interface	Methods	Description
ActionEvent	ActionListener	public void actionPerformed(ActionEvent ae)	This method is invoked when a button is pressed or a list item has been double clicked by the user.
ItemEvent	ItemListener	public void itemStateChanged(ItemEvent e)	This method is invoked when an item such as checkbox / list item is selected or deselected by the user.
MouseEvent	MouseListener	public void mouseEntered(MouseEvent me)	This method is called when a mouse cursor enters a window listening for MouseEvent.
		public void mousePressed(MouseEvent me)	This method is called when a mouse button is being pressed within a window listening for MouseEvent.
		public void mouseClicked(MouseEvent me)	This method is called when a mouse button is clicked within a window listening for MouseEvent.
		public void mouseReleased(MouseEvent me)	This method is called when a mouse button is released within a window listening for MouseEvent.
		public void mouseExited(MouseEvent me)	This method is called when a mouse cursor exits a window listening for MouseEvent.
KeyEvent	KeyListener	public void keyPressed(KeyEvent ke)	This method is called when a key is pressed on the keyboard.
		public void keyTyped(KeyEvent ke)	This method is called when pressing a key on the keyboard has resulted in a character.
		public void keyReleased(KeyEvent ke)	This method is called when a key is released on the keyboard.

## **PROGRAMS:**

### **Question 1 –**

Write a Java program to create an AWT GUI for a Login page and handle the event.

### **CODE –**

```
package Practs4;
import java.awt.*;
import java.awt.event.*;

public class LoginGUI extends Frame implements ActionListener{

    Label label1, label2, label3, label4;
    TextField textField1, textField2;
    Button button1;

    public LoginGUI() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        label1 = new Label("Login");
        label2 = new Label("Username:");
        label3 = new Label("Password:");
        label4 = new Label("");
        textField1 = new TextField(20);
        textField2 = new TextField(20);
        button1 = new Button("Login");

        add(label1);
        add(label2);    add(textField1);
        add(label3);    add(textField2);
        add(button1);
        add(label4);

        button1.addActionListener(this);
        setVisible(true);
        setSize(500,400);
    }

    private void button1ActionPerformed(ActionEvent evt) {

        if(textField1.getText().equals("user1") && textField2.getText().equals("1234"))
        {
            label4.setText("LOGGED IN SUCCESSFULLY");
        }
    }
}
```

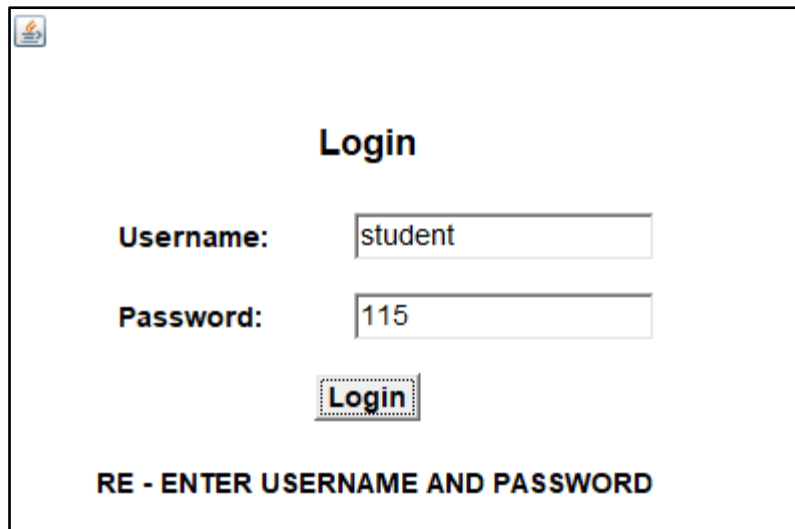
```

else
{
    label4.setText("RE - ENTER USERNAME AND PASSWORD");
}
}

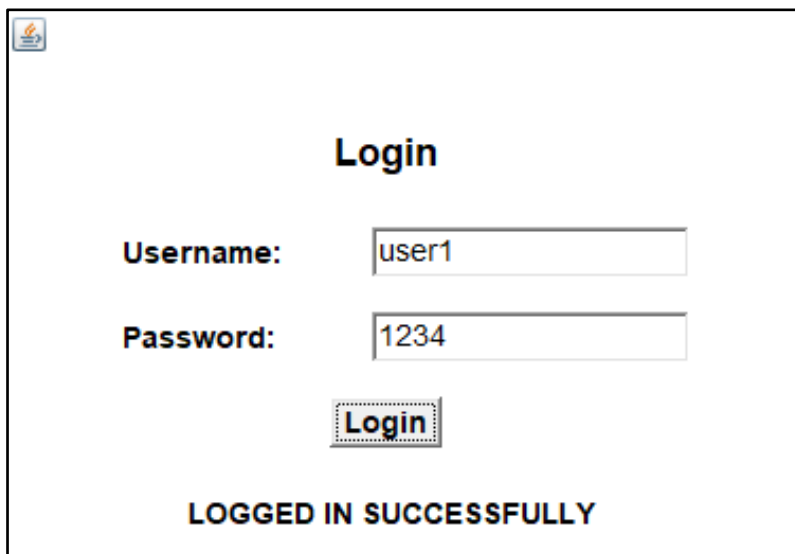
public static void main(String args[]) {
    LoginGUI obj = new LoginGUI();
}
}

```

**OUTPUT –**



**Figure 2:** Output for Login AWT GUI (Incorrect username and password)



**Figure 3:** Output for Login AWT GUI (Correct username and password)

**Question 2 –**

Write a Java program to create an AWT GUI for a Calculator and perform arithmetical operations.

**CODE –**

```
package Practs4;
import java.awt.*;
import java.awt.event.*;

public class Calculator extends Frame implements ActionListener{
    Label label1, label2, label3, label4, label 5, label6;
    TextField textField1, textField2, textField3;
    Button button1, button2, button3, button4, button5;

    public Calculator() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        label1 = new Label("Calculator");
        label2 = new Label("Enter first number:");
        label3 = new Label("Enter second number:");
        label4 = new Label("Answer:");
        label5 = new Label("");
        label6 = new Label("");
        textField1 = new TextField(20);
        textField2 = new TextField(20);
        textField3 = new TextField(20);
        button1 = new Button("ADD");
        button2 = new Button("SUBTRACT");
        button3 = new Button("MULTIPLY");
        button4 = new Button("DIVIDE");
        button5 = new Button("CLEAR");

        add(label1);
        add(label2);      add(textField1);
        add(label5);
        add(label3);      add(textField2);
        add(label6);
        add(label4);      add(textField3);
        add(button1);      add(button2);      add(button3);      add(button4);
        add(button5);

        button1.addActionListener(this);
        button2.addActionListener(this);
        button3.addActionListener(this);
```

```

        button4.addActionListener(this);
        button5.addActionListener(this);

        setVisible(true);
        setSize(500,400);
    }

    private void button1ActionPerformed(ActionEvent evt) {
        label5.setText("+");
        label6.setText("=");
        int num1, num2, ans;
        num1 = Integer.parseInt(textField1.getText());
        num2 = Integer.parseInt(textField2.getText());
        ans = num1 + num2;
        textField3.setText("" + ans);
    }

    private void button2ActionPerformed(ActionEvent evt) {
        label5.setText("-");
        label6.setText("=");
        int num1, num2, ans;
        num1 = Integer.parseInt(textField1.getText());
        num2 = Integer.parseInt(textField2.getText());
        ans = num1 - num2;
        textField3.setText("" + ans);
    }

    private void button3ActionPerformed(ActionEvent evt) {
        label5.setText("*");
        label6.setText("=");
        int num1, num2, ans;
        num1 = Integer.parseInt(textField1.getText());
        num2 = Integer.parseInt(textField2.getText());
        ans = num1 * num2;
        textField3.setText("" + ans);
    }

    private void button4ActionPerformed(ActionEvent evt) {
        label5.setText("/");
        label6.setText("=");
        float num1, num2, ans;
        num1 = Float.parseFloat(textField1.getText());
        num2 = Float.parseFloat(textField2.getText());
        ans = num1 / num2;
    }

```

```

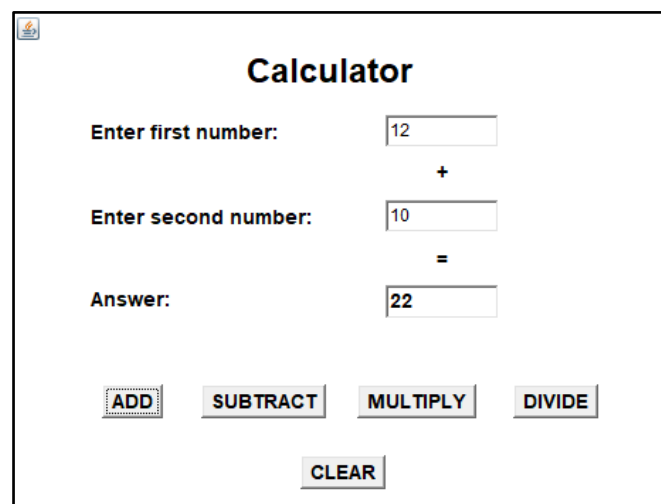
        textField3.setText("" + ans);
    }

    private void button5ActionPerformed(ActionEvent evt) {
        textField1.setText("");
        textField2.setText("");
        textField3.setText("");
        label5.setText("");
        label6.setText("");
    }

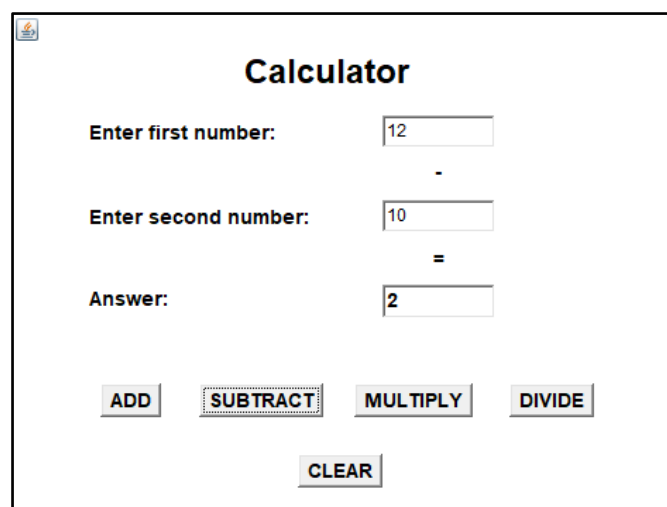
    public static void main(String args[]) {
        Calculator obj = new Calculator();
    }
}

```

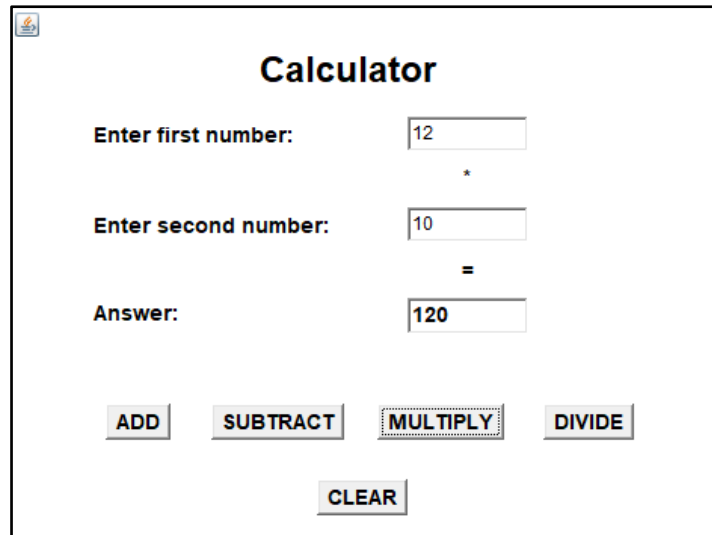
**OUTPUT –**



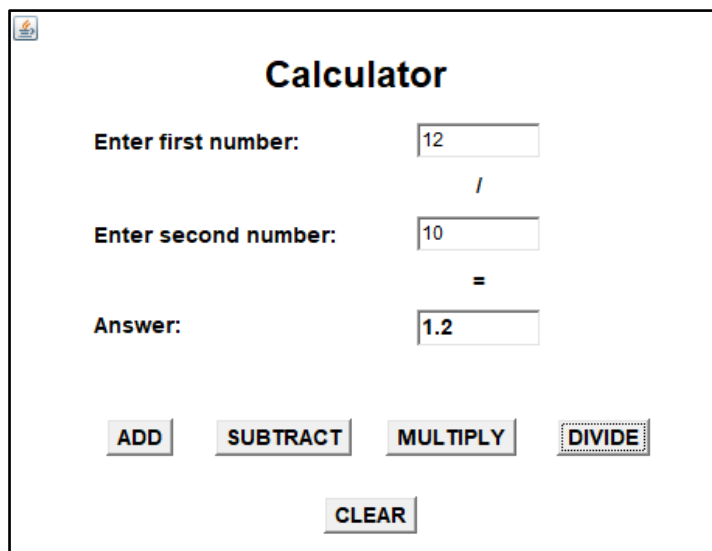
**Figure 4:** Output for Calculator AWT GUI (Mathematical operation: Addition)



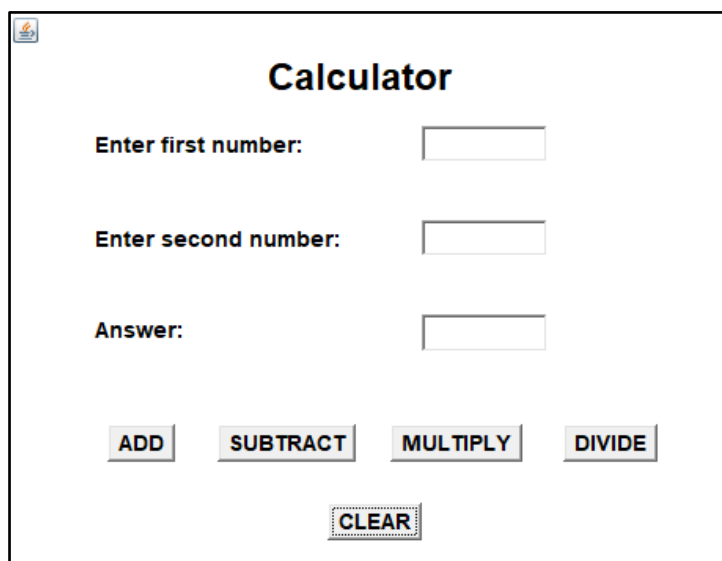
**Figure 5:** Output for Calculator AWT GUI (Mathematical operation: Subtraction)



**Figure 6:** Output for Calculator AWT GUI (Mathematical operation: Multiplication)



**Figure 7:** Output for Calculator AWT GUI (Mathematical operation: Division)



**Figure 8:** Output for Calculator AWT GUI (Clear button)

**Question 3 –**

Write a Java program to create an AWT GUI for accepting Student Details.

**CODE –**

```
package Practs4;
import java.awt.*;
import java.awt.event.*;

public class StudentDetails extends Frame implements ActionListener, ItemListener{
    Label label1, label2, label3, label4, label5;
    TextField textField1, textField2;
    Checkbox checkbox1, checkbox2, checkbox3;
    Button button1;

    public StudentDetails() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        label1 = new Label("STUDENT DETAILS");
        label2 = new Label("Name:");
        label3 = new Label("Contact No.:");
        label4 = new Label("Courses offered:");
        label5 = new Label("");
        textField1 = new TextField(20);
        textField2 = new TextField(20);
        checkbox1 = new Checkbox("Bioinformatics", false);
        checkbox2 = new Checkbox("Botany", false);
        checkbox3 = new Checkbox("Biochemistry", false);
        button1 = new Button("Submit");

        add(label1);
        add(label2);      add(textField1);
        add(label3);      add(textField2);
        add(label4);      add(checkbox1);      add(checkbox2);      add(checkbox3);
        add(button1);
        add(label5);

        button1.addActionListener(this);
        setVisible(true);
        setSize(500,400);
    }
}
```



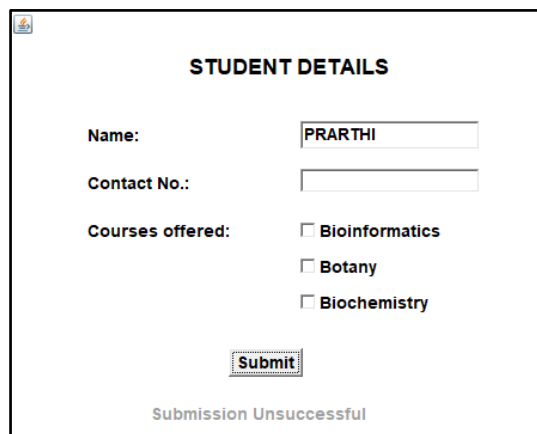
```

private void button1ActionPerformed(ActionEvent evt) {
    if(textField1.getText().equals("") || textField2.getText().equals("") ||
checkbox1.getState() == false || checkbox2.getState() == false || checkbox3.getState()
== false)
    {
        label5.setText("Submission Unsuccessful");
    }
    else
    {
        label5.setText("Submission Successful");
    }
}

public static void main(String args[]) {
    StudentDetails obj = new StudentDetails();
}
}

```

**OUTPUT –**



**STUDENT DETAILS**

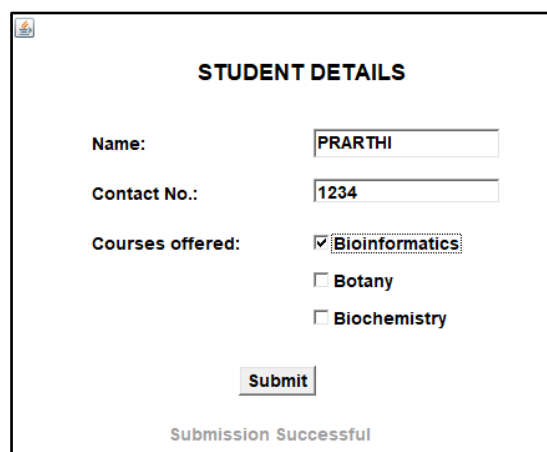
Name:

Contact No.:

Courses offered: ☐ Bioinformatics  
☐ Botany  
☐ Biochemistry

Submission Unsuccessful

**Figure 9:** Output for Student Details AWT GUI (Unsuccessful Submission due to provision of incomplete details)



**STUDENT DETAILS**

Name:

Contact No.:

Courses offered: ☒ Bioinformatics  
☐ Botany  
☐ Biochemistry

Submission Successful

**Figure 10:** Output for Calculator AWT GUI (Successful Submission with Complete details)

**Question 4 –**

Write a Java program to create an AWT GUI to display various mouse events.

**CODE –**

```
package Practs4;
import java.awt.*;
import java.awt.event.*;

public class MouseListenerGUI extends Frame implements MouseListener {
    TextField textField1;

    public MouseListenerGUI() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        textField1 = new TextField(100);
        textField1.addMouseListener();
        setVisible(true);
        setSize(500,400);
    }

    private void textField1MouseEntered(MouseEvent evt) {
        textField1.setText("MOUSE ENTERED IN TEXTFIELD1");
    }

    private void textField1MouseClicked(MouseEvent evt) {
        textField1.setText("MOUSE CLICKED IN TEXTFIELD1");
    }

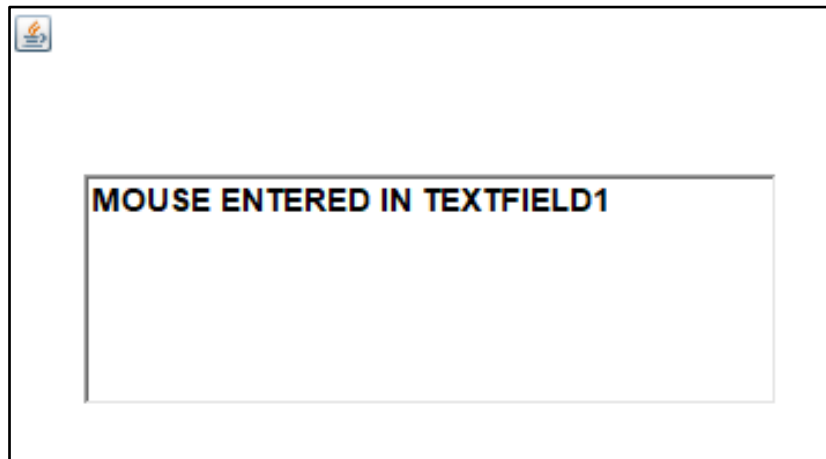
    private void textField1MouseExited(MouseEvent evt) {
        textField1.setText("MOUSE EXITED FROM TEXTFIELD1");
    }

    private void textField1MousePressed(MouseEvent evt) {
        textField1.setText("MOUSE PRESSED IN TEXTFIELD1");
    }

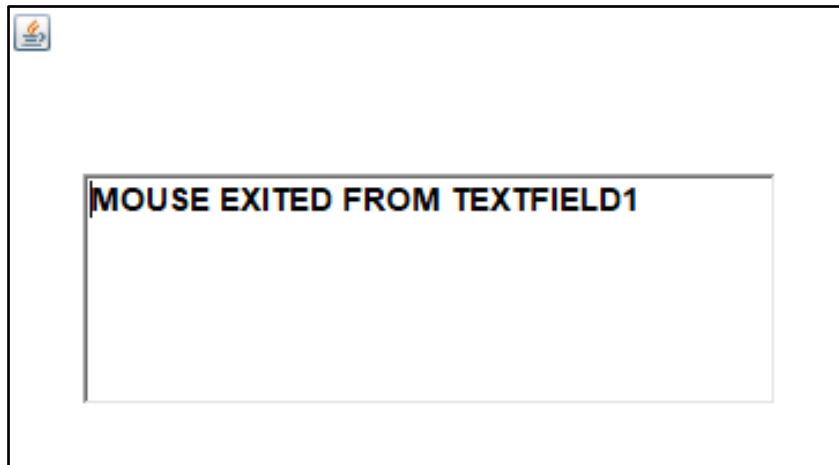
    private void textField1MouseReleased(MouseEvent evt) {
        textField1.setText("MOUSE RELEASED IN TEXTFIELD1");
    }

    public static void main(String args[]) {
        MouseListenerGUI obj = new MouseListenerGUI();
    }
}
```

## OUTPUT –



**Figure 11:** Output for AWT GUI for various mouse events (MouseEntered)



**Figure 12:** Output for AWT GUI for various mouse events (MouseExited)



**Figure 13:** Output for AWT GUI for various mouse events (MouseClicked)



**Figure 14:** Output for AWT GUI for various mouse events (MousePressed)

## **RESULTS:**

Using the Java programming language, the concept of Abstract Windows Toolkit (AWT) was explored and demonstrated in order to create AWT GUI for a login page, a calculator for performing arithmetical operations, a form for accepting details of a student and to display various mouse events.

## **CONCLUSION:**

The concept of Abstract Windows Toolkit (AWT) of Java programming language was demonstrated.

## **REFERENCES:**

1. *java.awt (Java Platform SE 8 )*. (2024, January 8).  
<https://docs.oracle.com/javase/8/docs/api/java/awt/package-summary.html>
  2. G. (2023, October 23). *Java AWT Toolkit*. GeeksforGeeks.  
<https://www.geeksforgeeks.org/java-awt-toolkit/>
  3. Dham, M. (2023, July 26). *Java AWT*. PrepBytes Blog.  
<https://www.prepbytes.com/blog/java/java-awt/>
  4. G. (2023, August 18). *Java AWT Tutorial*. GeeksforGeeks.  
<https://www.geeksforgeeks.org/java-awt-tutorial/>
-

## Practical 5 JAVA: SWING GUI

### AIM:

To explore and demonstrate the concept of SWING by creating GUI – based programs in Java programming language.

### INTRODUCTION:

Java is a versatile programming language released in 1995 that runs on various platforms without needing recompilation. Its object-oriented design and platform independence make it popular for diverse applications, from web development to mobile apps. Known for its "write once, run anywhere" capability, Java simplifies development and promotes code reusability across systems.

### SWING

Java Swing is a GUI widget toolkit for Java that offers a rich set of components for creating window-based applications. It is part of the Java Foundation Classes (JFC) and is entirely written in Java, providing platform-independent and lightweight components. Swing provides a more extensive range of User Interface (UI) components compared to AWT, with features like drag-and-drop support, pluggable look and feel, and advanced components such as tables, lists, scroll panes, and tabbed panes. Swing follows the Model-View-Controller (MVC) architecture, allowing for a clear separation of concerns in GUI development. The framework is known for its flexibility, asynchronous event handling, and the ability to override the native operating system's GUI rendering. Overall, Java Swing is a powerful tool for developing modern and complex graphical user interfaces in Java applications.

### Advantages of SWING

1. **Platform Independence:** Swing components are written entirely in Java, making them inherently platform-independent. The application will remain consistent across Windows, macOS, Linux, and other operating systems.
2. **Rich Functionality:** Swing offers a wider range of components and features compared to the earlier Abstract Window Toolkit (AWT), providing more flexibility and control over UI design.
3. **Pluggable Look and Feel:** Swing allows you to change the overall look and feel of the application to match the native appearance of the underlying operating system or create a custom theme. This enhances user familiarity and aesthetics.
4. **Event Handling:** Swing provides a powerful event handling mechanism that allows you to respond to user interactions with the UI components, making your application interactive.
5. **Model-View-Controller (MVC) Support:** Swing aligns well with the MVC design pattern, which promotes cleaner code separation and maintainability.

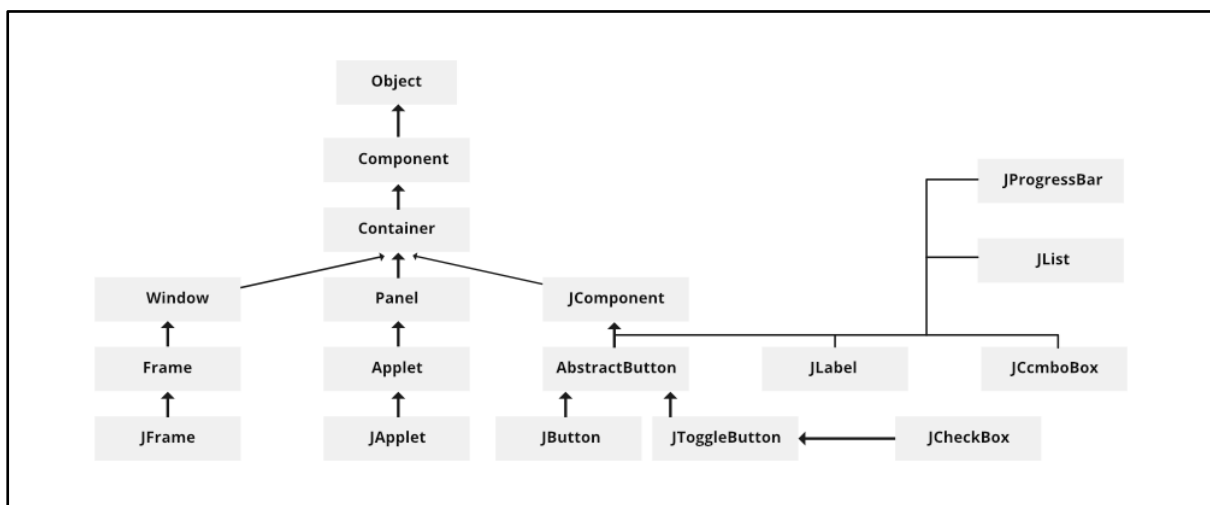
## Disadvantages of SWING

1. **Learning Curve:** While Swing is considered relatively easy to learn compared to some other GUI toolkits, it has a steeper learning curve than simpler options like JavaFX, especially for beginners.
2. **Verbosity:** Swing code can sometimes be more verbose than some more modern toolkits, requiring more lines of code to achieve the same functionality.
3. **Potential Performance Considerations:** In rare cases, Swing applications might have slightly slower performance compared to highly optimized toolkits, but this is usually not a significant concern for most desktop applications.

## Model-View-Controller (MVC) Architecture

Swing uses the model-view-controller architecture (MVC) as the fundamental design behind each of its components. Essentially, MVC breaks GUI components into three elements. Each of these elements plays a crucial role in how the component behaves.

1. **Model:** The model encompasses the state data for each component. There are different models for different types of components. For example, the model of a scrollbar component might contain information about the current position of its adjustable “thumb,” its minimum and maximum values, and the thumb’s width (relative to the range of values). Note that this information remains the same no matter how the component is painted on the screen; model data always exists independent of the component’s visual representation.
2. **View:** The view refers to how you see the component on the screen.
3. **Controller:** The controller is the portion of the user interface that dictates how the component interacts with events. Events come in many forms — a mouse click, gaining or losing focus, a keyboard event that triggers a specific menu command, or even a directive to repaint part of the screen. The controller decides how each component will react to the event—if it reacts at all.



**Figure 1:** Hierarchy of SWING Components

## **PROGRAMS:**

### **Question 1 –**

Write a program to create a SWING GUI and handle the event by displaying the dialog box when an item is selected.

### **CODE –**

```
package Practs5;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class CountryList extends javax.swing.JFrame implements ActionListener{

    JLabel jLabel1;
    JComboBox jComboBox1;
    JButton jButton1;

    String countrylist[] = {"Afghanistan", "Albania", "Algeria", "Brazil", "Brunei",
"Bulgaria", "Cabo Verde", "Cambodia", "Cameroon", "Canada", "Denmark",
"Djibouti", "Dominica", "Egypt", "El Salvador", "Equatorial Guinea", "Fiji",
"Finland", "France", "Germany", "Ghana", "Greece", "Haiti", "Honduras", "Hungary",
"Iceland", "India", "Indonesia", "Jamaica", "Japan", "Jordan", "Kazakhstan", "Kenya",
"Korea, South", "Laos", "Latvia", "Lebanon", "Madagascar", "Malawi", "Malaysia",
"Namibia", "Nauru", "Nepal", "Oman", "Palau", "Panama", "Qatar", "Romania",
"Russia", "Rwanda", "Saint Kitts and Nevis", "Saint Lucia", "Saint Vincent and the
Grenadines", "Taiwan", "Tajikistan", "Tanzania", "Uganda", "Ukraine", "United Arab
Emirates", "Vanuatu", "Vatican City", "Venezuela", "Vietnam", "Yemen", "Zambia",
"Zimbabwe" };

    public CountryList() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        jLabel1 = new JLabel("Select Country: ");
        jComboBox1 = new JComboBox(countrylist);
        jButton1 = new JButton("Submit");

        add(jLabel1);    add(jComboBox1);    add(jButton1);

        jButton1.addActionListener(this);
        setVisible(true);
        setSize(500,400);
    }
}
```

```

private void jButton1ActionPerformed(ActionEvent evt)
{
    String selected_country = "" + jComboBox1.getSelectedItem();
    JOptionPane jOptionPane1 = new JOptionPane();
    jOptionPane.showMessageDialog(jOptionPane1, "" + selected_country + "
has been selected");
}

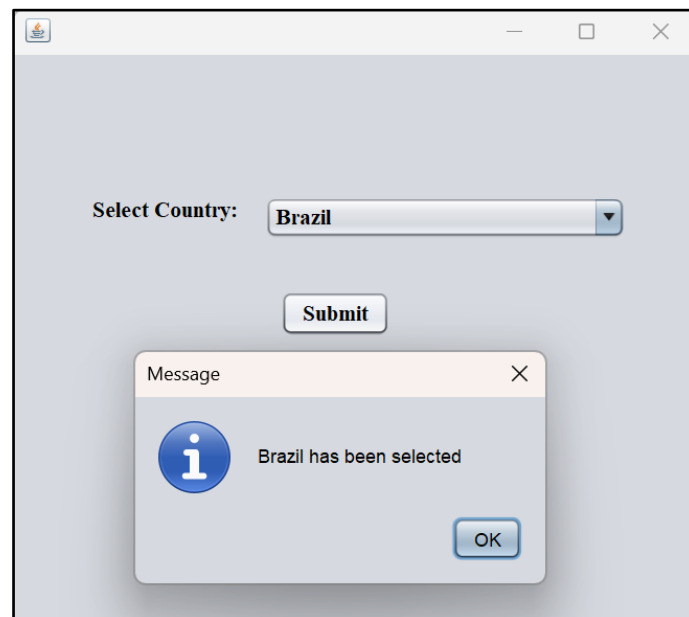
public static void main(String args[]) {
    CountryList obj = new CountryList();
}
}

```

**OUTPUT –**



**Figure 2:** GUI for selecting a country from the provided list



**Figure 3:** Output in Dialog box when 'Brazil' is selected and 'Submit' button is clicked



**Question 2 –**

Write a program to create a SWING GUI and handle the event by displaying the dialog box when an item is selected.

**CODE –**

```
package Practs5;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class AnimalSelection extends JFrame implements
ActionListener{

    JLabel jLabel1;
    JList jList1;
    JScrollPane jScrollPane1;
    JButton jButton1;

    String animals[] = {"Bird", "Cat", "Dog", "Rabbit"};

    public AnimalSelection() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        jLabel1 = new JLabel("Select Animal: ");
        jList1 = new JList(animals);
        jScrollPane1 = new JScrollPane(jList1);
        jButton1 = new JButton("Submit");

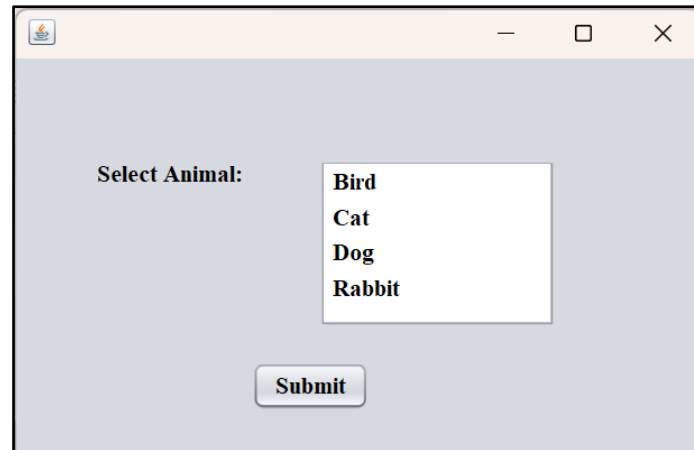
        add(jLabel1);    add(jScrollPane1);    add(jButton1);

        jButton1.addActionListener(this);
        setVisible(true);
        setSize(500,400);
    }

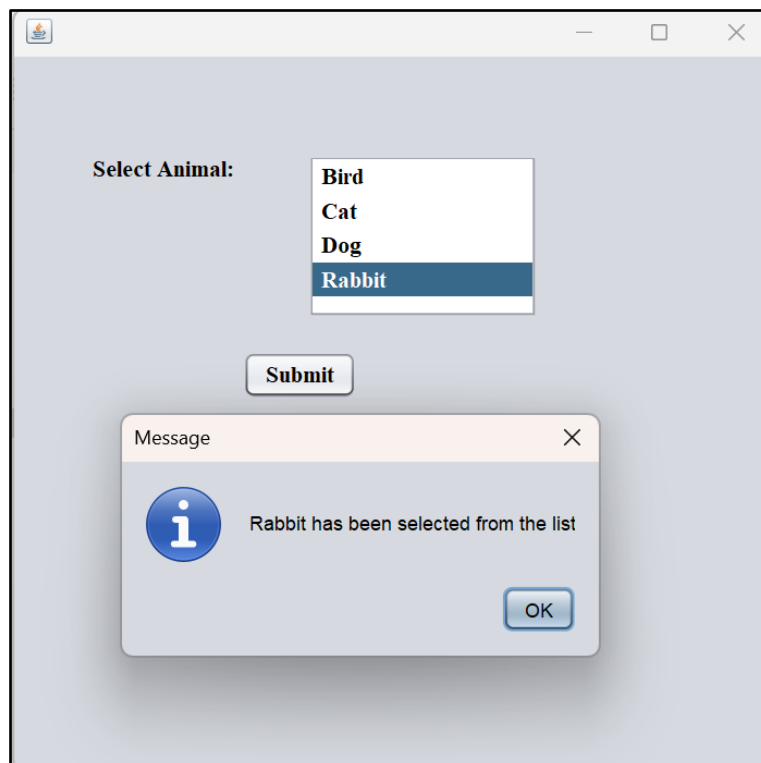
    private void jButton1ActionPerformed(ActionEvent evt)
    {
        String selected_animal = jList1.getSelectedValue();
        JOptionPane JOptionPane1 = new JOptionPane();
        JOptionPane.showMessageDialog(JOptionPane1, "" + selected_animal + "
has been selected from the list");
    }
}
```

```
public static void main(String args[]) {  
    AnimalSelection obj = new AnimalSelection();  
}  
}
```

**OUTPUT –**



**Figure 4:** GUI for selecting an animal from the provided list



**Figure 5:** Output in Dialog box when 'Rabbit' is selected and 'Submit' button is clicked

**Question 3 –**

Write a program to create a SWING GUI and handle the event by displaying the dialog box when an item is selected.

**CODE –**

```
package Practs5;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class FoodSelection extends JFrame implements ActionListener{

    JLabel jLabel1;
    JRadioButton jRadioButton1, jRadioButton2, jRadioButton3;
    ButtonGroup buttonGroup1;
    JButton jButton1;

    public FoodSelection() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        jLabel1 = new JLabel("Select Food: ");

        jRadioButton1 = new JRadioButton();
        jRadioButton2 = new JRadioButton();
        jRadioButton3 = new JRadioButton();

        jRadioButton1.setText("French Fries");
        jRadioButton2.setText("Onion Rings");
        jRadioButton3.setText("Ice-cream");

        buttonGroup1 = new ButtonGroup();
        buttonGroup1.add(jRadioButton1);
        buttonGroup1.add(jRadioButton2);
        buttonGroup1.add(jRadioButton3);

        jButton1 = new JButton("Submit");

        add(jLabel1);
        add(jRadioButton1);    add(jRadioButton2);    add(jRadioButton3);
        add(jButton1);

        jButton1.addActionListener(this);
        setVisible(true);
        setSize(500,400);
    }
}
```

```

private void jButton1ActionPerformed(ActionEvent evt)
{
    JOptionPane jOptionPane1 = new JOptionPane();

    if (jRadioButton1.isSelected() == true){
        jOptionPane.showMessageDialog(jOptionPane1, "French Fries has
        been selected");
    }

    else if (jRadioButton2.isSelected() == true){
        jOptionPane.showMessageDialog(jOptionPane1, "Onion Rings has
        been selected");
    }

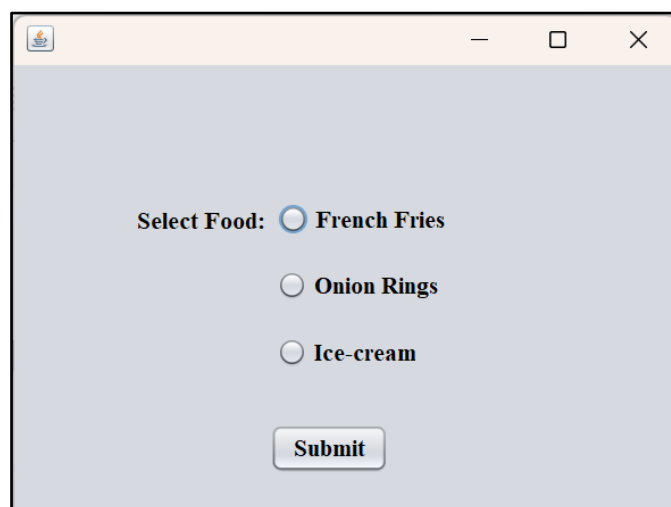
    else if (jRadioButton3.isSelected() == true){
        jOptionPane.showMessageDialog(jOptionPane1, "Ice-cream has
        been selected");
    }

    else {
        jOptionPane.showMessageDialog(jOptionPane1, "No Option has
        been selected");
    }
}

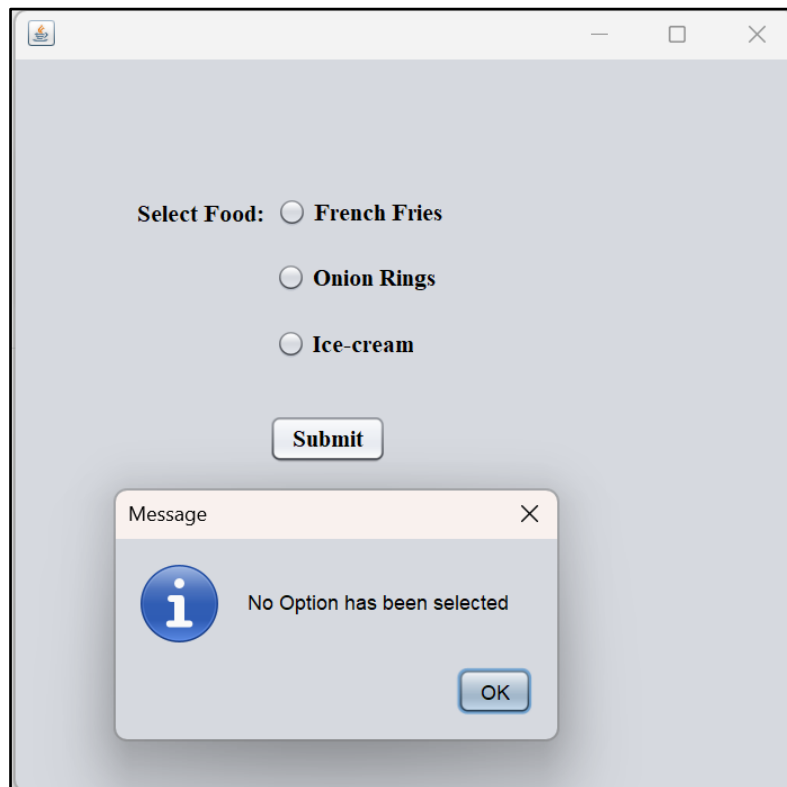
public static void main(String args[]) {
    FoodSelection obj = new FoodSelection();
}
}

```

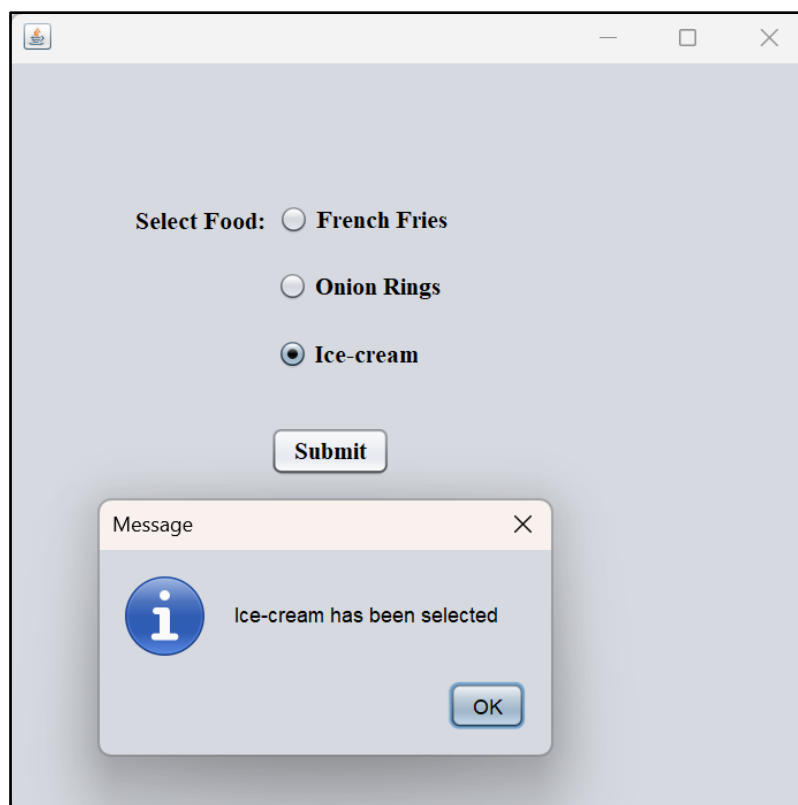
**OUTPUT –**



**Figure 6:** GUI for selecting a food item



**Figure 7:** Output in Dialog box when no option has been selected and ‘Submit’ button is clicked



**Figure 8:** Output in Dialog box when ‘Ice-cream’ is selected and ‘Submit’ button is clicked

**Question 4 –**

Write a program to create a SWING GUI and handle the event by displaying the dialog box when an item is selected.

**CODE –**

```
package Practs5;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class LoginForm extends JFrame implements ActionListener{

    JLabel jLabel1, jLabel2, jLabel3;
    JTextField jTextField1, jTextField2;
    JButton jButton1;

    public LoginForm() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        jLabel1 = new JLabel("Login Form ");
        jLabel2 = new JLabel("Username: ");
        jLabel3 = new JLabel("Password: ");
        jTextField1 = new JTextField(20);
        jTextField2 = new JTextField(20);
        jButton1 = new JButton("Click here to Login");

        add(jLabel1);
        add(jLabel2);    add(jTextField1);
        add(jLabel3);    add(jTextField2);
        add(jButton1);

        jButton1.addActionListener(this);
        setVisible(true);
        setSize(500,400);
    }

    private void jButton1ActionPerformed(ActionEvent evt)
    {
        JOptionPane jPanel1 = new JOptionPane();
        String username = jTextField1.getText();
        String password = jTextField2.getText();

        if(username.equals("user1") && password.equals("1234")){
            JOptionPane.showMessageDialog(jPanel1, "Login Successful");
        }
    }
}
```

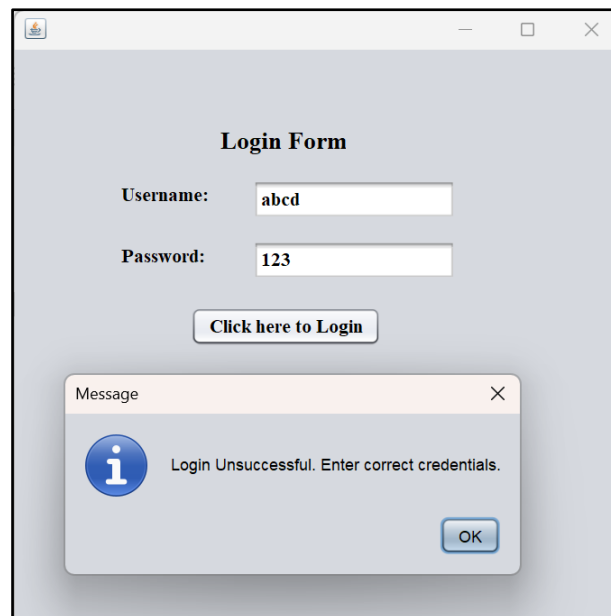
```

else {
    JOptionPane.showMessageDialog(jOptionPane1, "Login Unsuccessful. Enter
correct credentials.");
}
}

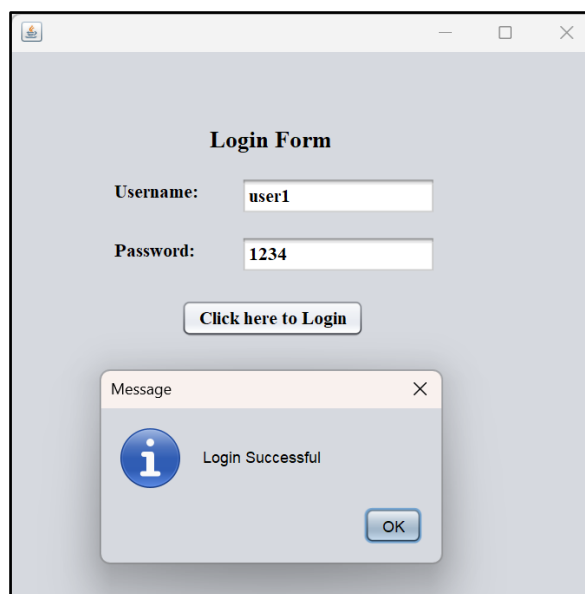
public static void main(String args[]) {
    LoginForm obj = new LoginForm();
}
}

```

**OUTPUT –**



**Figure 9:** GUI for Login Form: Output for entering incorrect username and password



**Figure 10:** GUI for Login Form: Output for entering correct username and password

**Question 5 –**

Write a program to create a SWING GUI and insert an image. Handle the event by displaying the dialog box when an item is selected.

**CODE –**

```
package Practs5;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class RegistrationForm extends JFrame implements
ActionListener{

    JLabel jLabel1, jLabel2, jLabel3, jLabel4, jLabel5, jLabel6;
    JTextField jTextField1, jTextField2, jTextField3;
    JRadioButton jRadioButton1, jRadioButton2, jRadioButton3;
    ButtonGroup buttonGroup1;
    JButton jButton1;

    public RegistrationForm() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        this.getContentPane().setBackground(new java.awt.Color(102,255,255));

        jLabel1 = new JLabel("");
        jLabel1.setIcon(new ImageIcon(getClass().getResource("/Practs5/img.png")));
        jLabel2 = new JLabel("Registration Form: ");
        jLabel3 = new JLabel("Enrollment Number: ");
        jLabel4 = new JLabel("Name: ");
        jLabel5 = new JLabel("Course: ");
        jLabel6 = new JLabel("Gender: ");

        jTextField1 = new JTextField(20);
        jTextField2 = new JTextField(20);
        jTextField3 = new JTextField(20);

        jRadioButton1 = new JRadioButton();
        jRadioButton2 = new JRadioButton();

        jRadioButton1.setText("Female");
        jRadioButton2.setText("Male");

        buttonGroup1 = new ButtonGroup();
        buttonGroup1.add(jRadioButton1);
        buttonGroup1.add(jRadioButton2);
```



```

jButton1 = new JButton("Register");

add(jLabel1);
add(jLabel2);
add(jLabel3);    add(jTextField1);
add(jLabel4);    add(jTextField2);
add(jLabel5);    add(jTextField3);
add(jLabel6);    add(jRadioButton1);    add(jRadioButton2);
add(jButton1);

jButton1.addActionListener(this);
setVisible(true);
setSize(500,400);
}

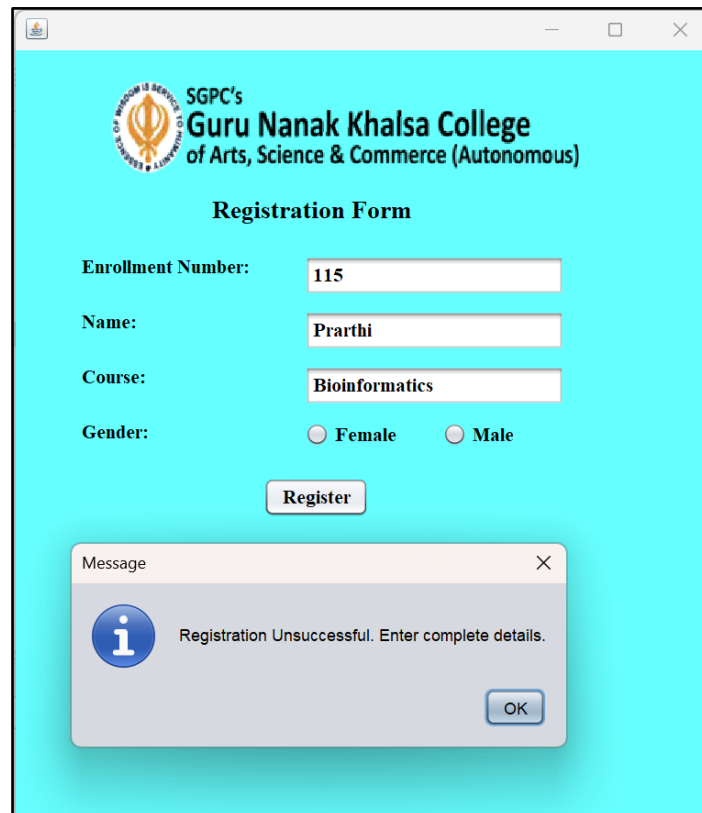
private void jButton1ActionPerformed(ActionEvent evt)
{
    JOptionPane jOptionPane1 = new JOptionPane();
    String enrollmentno = jTextField1.getText();
    String name = jTextField2.getText();
    String course = jTextField3.getText();

    if (enrollmentno.equals("") || name.equals("") || course.equals("") ||
        (jRadioButton1.isSelected() == false && jRadioButton2.isSelected() ==
        false)){
        JOptionPane.showMessageDialog(jOptionPane1, "Registration
        Unsuccessful. Enter complete details.");
    }
    else {
        JOptionPane.showMessageDialog(jOptionPane1, "Registration
        Successful.");
    }
}

public static void main(String args[]) {
    RegistrationForm obj = new RegistrationForm();
}
}

```

## OUTPUT –

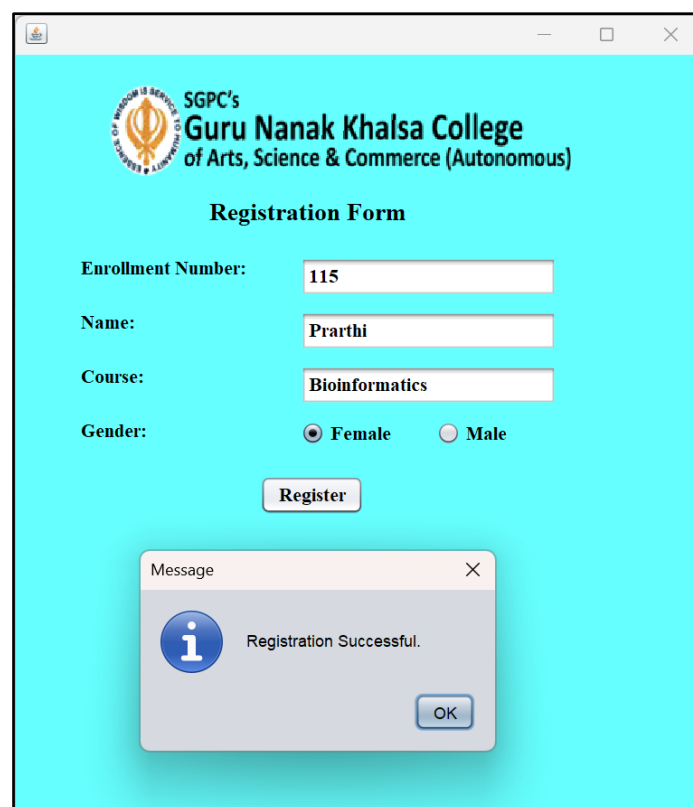


The screenshot displays a web application window titled "SGPC's Guru Nanak Khalsa College of Arts, Science & Commerce (Autonomous)". The main heading is "Registration Form". The form contains the following fields and controls:

- Enrollment Number:** A text input field containing the value "115".
- Name:** A text input field containing the value "Prarthi".
- Course:** A text input field containing the value "Bioinformatics".
- Gender:** Two radio button options: "Female" (which is selected) and "Male".
- Register:** A button located below the form fields.

A modal message box is open in the foreground, titled "Message". It contains an information icon and the text "Registration Unsuccessful. Enter complete details." with an "OK" button at the bottom right.

**Figure 11:** GUI for Registration Form: Output for entering incomplete details



The screenshot displays the same web application window as Figure 11. The form fields are identical: Enrollment Number "115", Name "Prarthi", Course "Bioinformatics", and Gender "Female" selected. The "Register" button is present.

A modal message box is open in the foreground, titled "Message". It contains an information icon and the text "Registration Successful." with an "OK" button at the bottom right.

**Figure 12:** GUI for Registration Form: Output for entering complete details

**Question 6 –**

Write a program to create a SWING GUI and handle the event by displaying the dialog box when an item is selected.

**CODE –**

```
package Practs5;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class StudentDetails extends JFrame implements ActionListener {

    JLabel jLabel1, jLabel2, jLabel3, jLabel4, jLabel5;
    JTextField jTextField1, jTextField2;
    JCheckBox jCheckBox1, jCheckBox2, jCheckBox3;
    ButtonGroup buttonGroup1;
    JButton jButton1;

    public StudentDetails() {
        AbsoluteLayout al = new AbsoluteLayout();
        setLayout(al);
        this.getContentPane().setBackground(new java.awt.Color(102,255,255));

        jLabel1 = new JLabel("Student Details");
        jLabel2 = new JLabel("Name: ");
        jLabel3 = new JLabel("Contact No.: ");
        jLabel4 = new JLabel("Courses Offered: ");
        jLabel5 = new JLabel("");

        jTextField1 = new JTextField(20);
        jTextField2 = new JTextField(20);
        jTextField3 = new JTextField(20);

        jCheckBox1 = new JCheckBox();
        jCheckBox2 = new JCheckBox();
        jCheckBox3 = new JCheckBox();

        jCheckBox1.setText("Bioinformatics");
        jCheckBox2.setText("Botany");
        jCheckBox3.setText("Chemistry");

        buttonGroup1 = new ButtonGroup();
        buttonGroup1.add(jCheckBox1);
        buttonGroup1.add(jCheckBox2);
        buttonGroup1.add(jCheckBox3);
```

```

jButton1 = new JButton("Submit");

add(jLabel1);
add(jLabel2);    add(jTextField1);
add(jLabel3);    add(jTextField2);
add(jLabel4);    add(jCheckBox1);  add(jCheckBox2);  add(jCheckBox3);
add(jButton1);
add(jLabel5);

jButton1.addActionListener(this);
setVisible(true);
setSize(500,400);
}

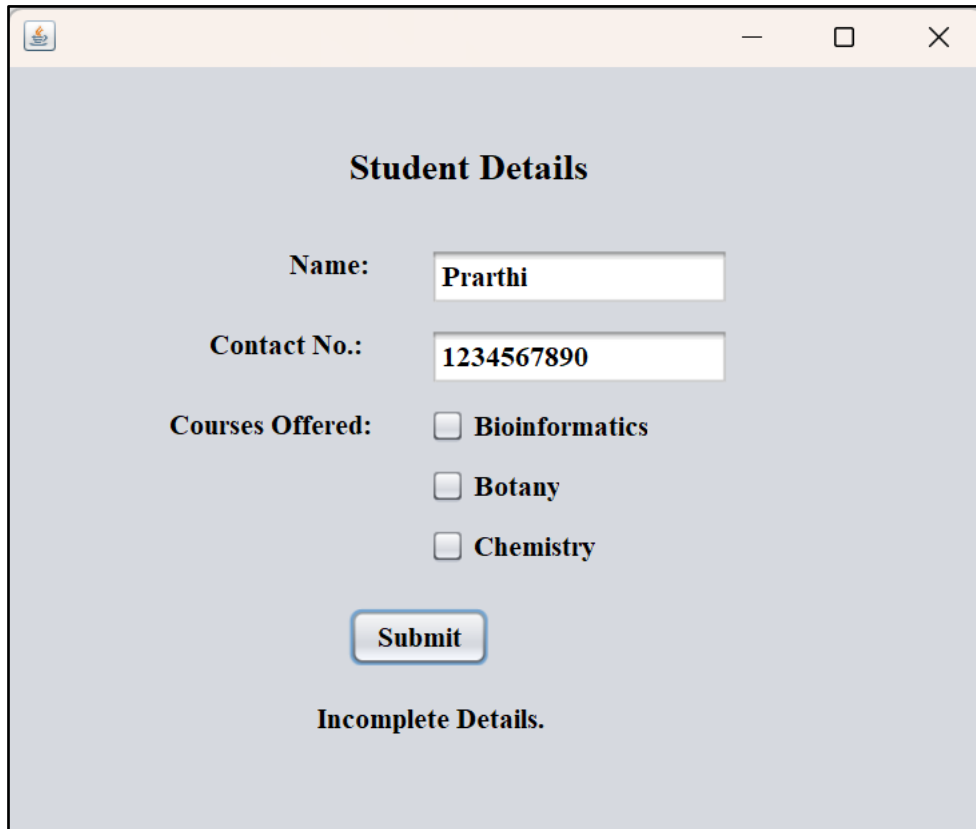
private void jButton1ActionPerformed(ActionEvent evt)
{
    JOptionPane jOptionPane1 = new JOptionPane();
    String name = jTextField1.getText();
    String contact = jTextField2.getText();

    if (name.equals("") || contact.equals("") || (jCheckBox1.isSelected() == false
    && jCheckBox2.isSelected() == false && jCheckBox3.isSelected() ==
    false)){
        jLabel5.setText("Incomplete Details.");
    }
    else {
        jLabel5.setText("Form Submitted Successfully");
    }
}

public static void main(String args[]) {
    StudentDetails obj = new StudentDetails();
}
}

```

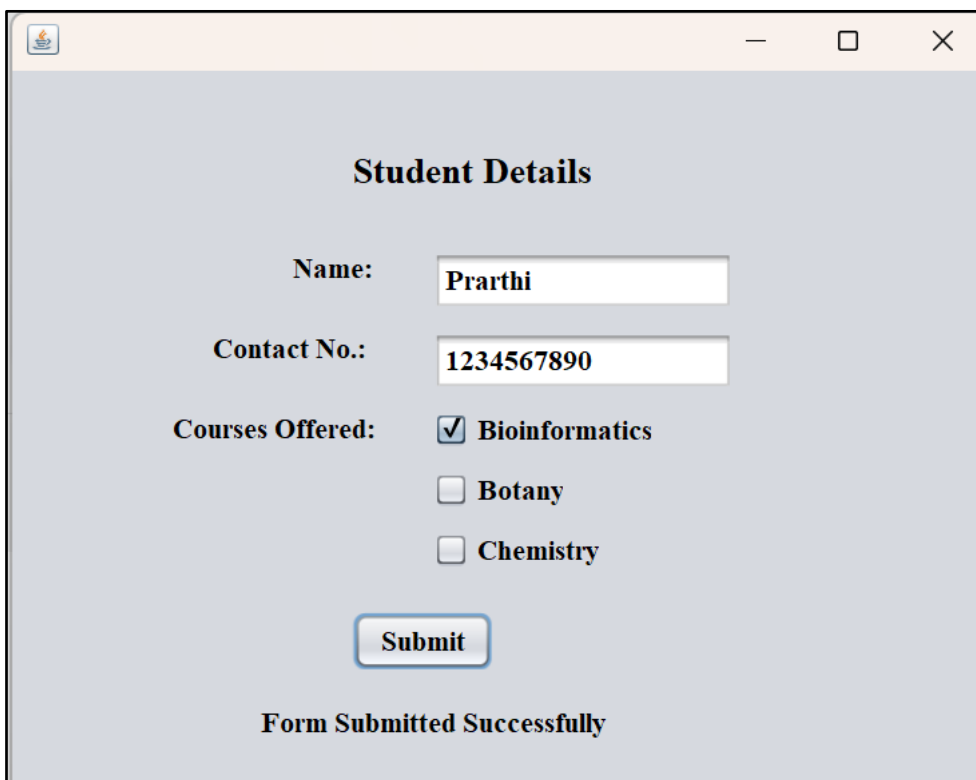
## OUTPUT –



A screenshot of a Java Swing window titled "Student Details". The window has a light gray background and a standard title bar with a minimize button, a maximize button, and a close button. The form contains the following elements:

- Name:** A text input field containing the text "Prarthi".
- Contact No.:** A text input field containing the text "1234567890".
- Courses Offered:** A group of three checkboxes, all of which are unchecked:
  - ☐ Bioinformatics
  - ☐ Botany
  - ☐ Chemistry
- Submit:** A rectangular button with a blue border and the text "Submit".
- Incomplete Details.** A text label at the bottom of the form.

**Figure 13:** GUI for Student Details: Output for entering incomplete details



A screenshot of the same Java Swing window titled "Student Details". The form contains the following elements:

- Name:** A text input field containing the text "Prarthi".
- Contact No.:** A text input field containing the text "1234567890".
- Courses Offered:** A group of three checkboxes, with the first one checked:
  - ☒ Bioinformatics
  - ☐ Botany
  - ☐ Chemistry
- Submit:** A rectangular button with a blue border and the text "Submit".
- Form Submitted Successfully** A text label at the bottom of the form.

**Figure 14:** GUI for Student Details: Output for entering complete details

## **RESULTS:**

Using the Java programming language, the concept of SWING was explored and demonstrated in order to create GUI for handling the events of selecting a country, an animal and a food item from a list and submissions of a login form, a registration form and a form for entering details of a student.

## **CONCLUSION:**

The concept of SWING was demonstrated using Java programming language.

## **REFERENCES:**

1. *Java Swing*. (n.d.). O'Reilly Online Learning. <https://www.oreilly.com/library/view/java-swing/156592455X/ch01s04.html>
  2. *MVC Architecture in Java - Javatpoint*. (n.d.). www.javatpoint.com. <https://www.javatpoint.com/mvc-architecture-in-java>
  3. *javax.swing (Java Platform SE 7 )*. (2020, June 24). <https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/javax/swing/package-summary.html>
  4. G. (2024, January 9). *Introduction to Java Swing*. GeeksforGeeks. <https://www.geeksforgeeks.org/introduction-to-java-swing/>
-

## **Practical 6**

### **JAVA: JDBC GUI**

#### **AIM:**

To explore and demonstrate the concept of Java Database Connectivity (JDBC) in Java programming language.

#### **INTRODUCTION:**

Java is a versatile programming language released in 1995 that runs on various platforms without needing recompilation. Its object-oriented design and platform independence make it popular for diverse applications, from web development to mobile apps. Known for its "write once, run anywhere" capability, Java simplifies development and promotes code reusability across systems.

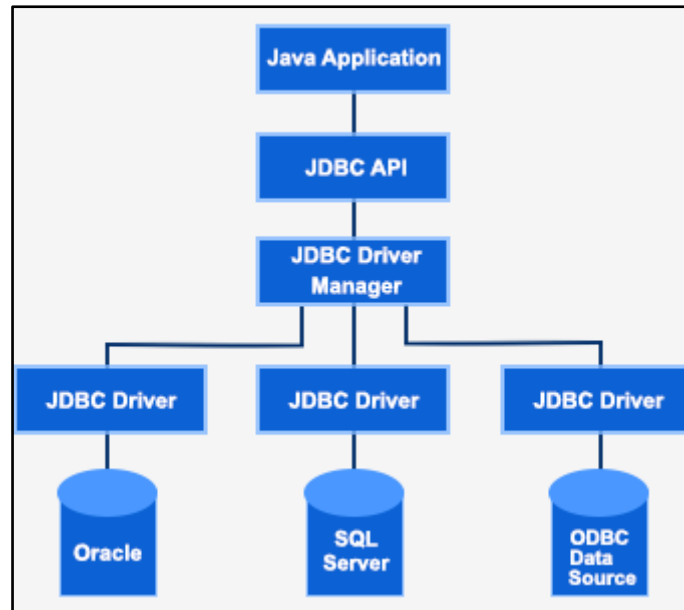
#### **Java Database Connectivity (JDBC)**

An essential Java API for handling result sets from databases, running queries, and connecting to databases is Java JDBC (Java Database Connectivity). Since its initial release in 1997 as a component of JDK 1.1, JDBC has developed to offer a standardized interface for Java applications to communicate with different kinds of data storage devices. The JDBC API serves as an interface between databases and Java programs, allowing programmers to send and receive SQL commands and queries, access and edit data records, and provide dependable database connectivity.

#### **JDBC Architecture**

JDBC consists of various components that each have a distinct function in the process of database connectivity, such as –

- 1. JDBC API** – provides various methods and interfaces for easy communication with the database.
- 2. JDBC Driver Manager** – loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.
- 3. JDBC Test Suite** – used to test the operation (such as insertion, deletion, updation) being performed by JDBC Drivers.
- 4. JDBC-ODBC Bridge** – connects database drivers to the database. This bridge translates the JDBC method call to the ODBC function call.

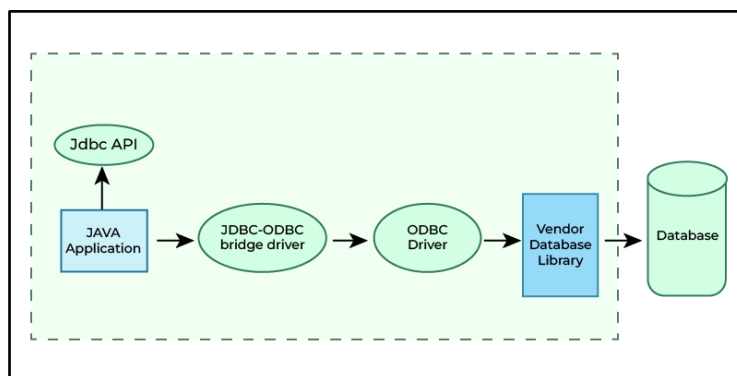


**Figure 1: JDBC Architecture**

### Types of JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand.

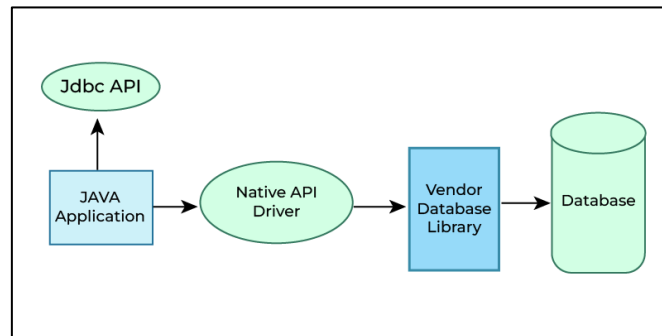
Type of JDBC Driver	Description	Advantages	Disadvantages
<b>JDBC-ODBC bridge driver – Type 1 driver</b>	It uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.	<ol style="list-style-type: none"> <li>1. This driver software is built-in with JDK so no need to install separately.</li> <li>2. It is a database independent driver.</li> </ol>	<ol style="list-style-type: none"> <li>1. As a common driver is used in order to interact with different databases, the data transferred through this driver is not so secured.</li> </ol>



**Figure 2: Type 1 Driver**

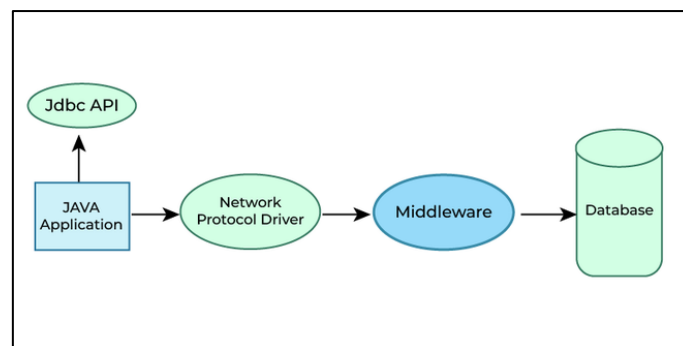


<b>Native-API driver – Type 2 driver</b>	<p>It uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to interact with different database, this driver needs their local API.</p>	<ol style="list-style-type: none"> <li>1. Native-API driver gives better performance than JDBC-ODBC bridge driver.</li> </ol>	<ol style="list-style-type: none"> <li>1. Driver needs to be installed separately in individual client machines.</li> <li>2. It is a database dependent driver.</li> </ol>
--	---	---	--



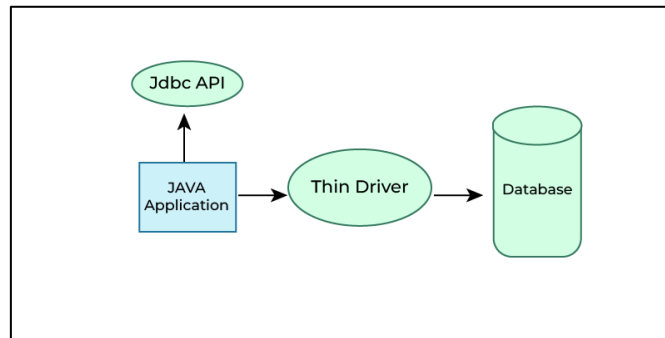
**Figure 3: Type 2 Driver**

<b>Network Protocol driver – Type 3 driver</b>	<p>It uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol.</p>	<ol style="list-style-type: none"> <li>1. Type-3 drivers are fully written in Java, hence they are portable drivers.</li> <li>2. No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.</li> </ol>	<ol style="list-style-type: none"> <li>1. Network support is required on client machine.</li> <li>2. Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.</li> </ol>
--	--	--	---



**Figure 4: Type 3 Driver**

<b>Thin driver – Type 4 driver</b>	This driver interacts directly with database. It does not require any native database library.	<ol style="list-style-type: none"> <li>1. It does not require any native library and middleware server, so no client-side or server-side installation.</li> <li>2. It is fully written in Java language, hence they are portable drivers.</li> </ol>	<ol style="list-style-type: none"> <li>1. If the database varies, then the driver will carry because it is database dependent.</li> </ol>
------------------------------------	--	--	---



**Figure 5:** Type 4 Driver

## **PROGRAMS:**

### **Question 1 –**

Create a JDBC for the following SWING Application GUI.

### **CODE –**

```
package SWINGJDBC_Practs6;
import java.sql.*;
import javax.swing.*;

public class NewUserRegister extends javax.swing.JFrame implements
ActionListener{

    JLabel jLabel1, jLabel2, jLabel3, jLabel4, jLabel5, jLabel6, jLabel7;
    JTextField jTextField1, jTextField2, jTextField3, jTextField4, jTextField5,
jTextField6;
    JButton jButton1;

    public NewUserRegister() {
        AbsoluteLayout a1 = new AbsoluteLayout();
        setLayout(a1);
        jLabel1 = new JLabel("New User Registration");
        jLabel2 = new JLabel("First Name:");
        jLabel3 = new JLabel("Last Name:");
        jLabel4 = new JLabel("Email Address:");
        jLabel5 = new JLabel("Username:");
        jLabel6 = new JLabel("Password:");
        jLabel7 = new JLabel("Mobile No.: ");

        jTextField1 = new JTextField(20);
        jTextField2 = new JTextField(20);
        jTextField3 = new JTextField(20);
        jTextField4 = new JTextField(20);
        jTextField5 = new JTextField(20);
        jTextField6 = new JTextField(20);

        jButton1 = new JButton("Register");

        add(jLabel1);
        add(jLabel2);    add(jTextField1);
        add(jLabel3);    add(jTextField2);
        add(jLabel4);    add(jTextField3);
        add(jLabel5);    add(jTextField4);
        add(jLabel6);    add(jTextField5);
```

```

add(jLabel7);    add(jTextField6);
add(jButton1);

jButton1.addActionListener(this);
setVisible(true);
setSize(500,400);
}

private void jButton1ActionPerformed(ActionEvent evt) {
    try
    {
        String f_name = jTextField1.getText();
        String l_name = jTextField2.getText();
        String e_add = jTextField3.getText();
        String uname = jTextField4.getText();
        String pass = jTextField5.getText();
        String mobile = jTextField6.getText();
        JOptionPane jOptionPane1 = new JOptionPane();

        Class.forName("com.mysql.jdbc.Driver");
        System.out.println("Registered");
        Connection con;
        Statement smt;
        con=DriverManager.getConnection("jdbc:mysql://localhost/registration",
            "root", "root");
        System.out.println("Connection Successful");
        smt=con.createStatement();

        if(f_name.equals("") || l_name.equals("") || e_add.equals("") ||
            uname.equals("") || pass.equals("") || mobile.equals(""))
        {
            JOptionPane.showMessageDialog(jOptionPane1, "Registration
            Unsuccessful. Enter complete details.");
        }

        else
        {
            JOptionPane.showMessageDialog(jOptionPane1, "Registration
            Successful.");
            String sql = "INSERT INTO REG_DETAILS VALUES('" + f_name + "','"
                + l_name + "','" + e_add + "','" + uname + "','" + pass + "','" + mobile + "');";
            smt.executeUpdate(sql);
        }
    }
}

```

```

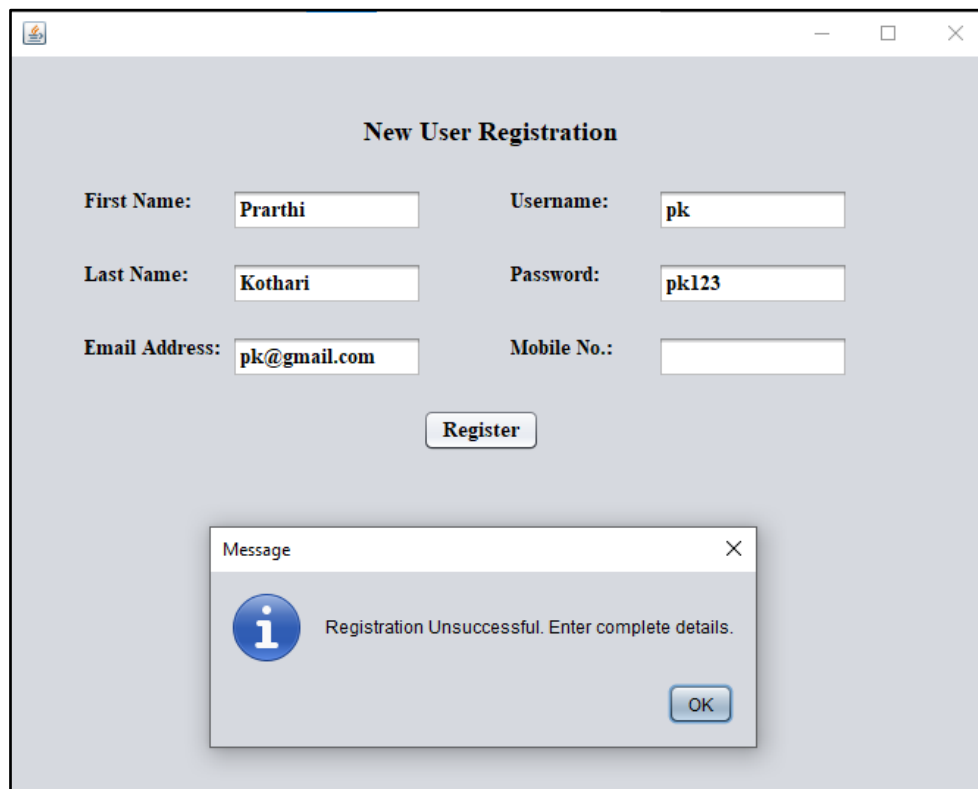
        smt.close();
        con.close();
    }

    catch(SQLException se)
    {
        se.printStackTrace();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public static void main(String args[]) {
    NewUserRegister obj = new NewUserRegister ();
}
}

```

## OUTPUT –



**Figure 6:** SWING GUI for New User Registration – Output for filling incomplete details

```
mysql> USE REGISTRATION;
Database changed
mysql> SELECT * FROM REG_DETAILS;
Empty set (0.00 sec)
```


**Figure 7:** Output for SQL – Displayed ‘Empty set’ due to incomplete details filled in the form

**New User Registration**

First Name:  Username:

Last Name:  Password:

Email Address:  Mobile No.:

Message  
 Registration Successful.

**Figure 8:** SWING GUI for New User Registration – Output for filling complete details

```
mysql> USE REGISTRATION;
Database changed
mysql> SELECT * FROM REG_DETAILS;
Empty set (0.00 sec)

mysql> SELECT * FROM REG_DETAILS;
+-----+-----+-----+-----+-----+-----+
| FIRST_NAME | LAST_NAME | EMAIL_ADDRESS | USERNAME | PASSWORD | MOBILE_NO |
+-----+-----+-----+-----+-----+-----+
| Prarthi    | Kothari   | pk@gmail.com  | pk       | pk123    | 1234567890 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

**Figure 9:** Output for SQL – Displayed the details filled in the form

**Question 2 –**

Create a JDBC for the following SWING Application GUI.

**CODE –**

```
package SWINGJDBC_Practs6;
import java.sql.*;
import java.awt.*;
import javax.swing.*;

public class LoginFormRegistration extends JFrame implements
ActionListener{

    JLabel jLabel1, jLabel2, jLabel3;
    JTextField jTextField1, jTextField2;
    JButton jButton1, jButton2;

    public LoginFormRegistration() {

        jLabel1 = new JLabel("Login Form");
        jLabel2 = new JLabel("Username:");
        jLabel3 = new JLabel("Password:");
        jTextField1 = new JTextField(20);
        jTextField2 = new JTextField(20);
        jButton1 = new JButton("Login");
        jButton2 = new JButton("Register");

        add(jLabel1);
        add(jLabel2);    add(jTextField1);
        add(jLabel3);    add(jTextField2);
        add(jButton1);    add(jButton2);

        jButton1.addActionListener(this);
        jButton2.addActionListener(this);

        setVisible(true);
        setSize(500,400);
    }

    private void jButton1ActionPerformed(ActionEvent evt) {
        try {
            String username = jTextField1.getText();
            String password = jTextField2.getText();
            String unamesql, upasssql;
            JOptionPane jOptionPane1 = new JOptionPane();
```

```

Class.forName("com.mysql.jdbc.Driver");
System.out.println("Registered");
Connection con;
Statement smt;
con=DriverManager.getConnection("jdbc:mysql://localhost/pract6", "root",
    "root");
System.out.println("Connection Successful");
smt = con.createStatement();

String sql = "SELECT * FROM REG_DETAILS WHERE USERNAME = '"
    + username + "'";
ResultSet rs = smt.executeQuery(sql);

if (rs.next()) {
    unamesql = rs.getString("USERNAME");
    upasssql = rs.getString("PASSWORD");

    if (username.equals(unamesql) && password.equals(upasssql)) {
        JOptionPane.showMessageDialog(jOptionPane1, "Login Successful.");
    }
    else {
        JOptionPane.showMessageDialog(jOptionPane1, "Credentials do not
            match.");
    }
}

else {
    JOptionPane.showMessageDialog(jOptionPane1, "User not found. Register
        to continue.");
}
rs.close();
smt.close();
con.close();
}

catch(SQLException se)
{
    se.printStackTrace();
}
catch(Exception e)
{
    e.printStackTrace();
}
}

```



```

private void jButton2ActionPerformed(ActionEvent evt) {
    try
    {
        String username = jTextField1.getText();
        String password = jTextField2.getText();
        JOptionPane JOptionPane1 = new JOptionPane();

        Class.forName("com.mysql.jdbc.Driver");
        System.out.println("Registered");
        Connection con;
        Statement smt;
        con=DriverManager.getConnection("jdbc:mysql://localhost/pract6", "root",
            "root");
        System.out.println("Connection Successful");
        smt=con.createStatement();
        JOptionPane.showMessageDialog(JOptionPane1, "Registration Successful.");

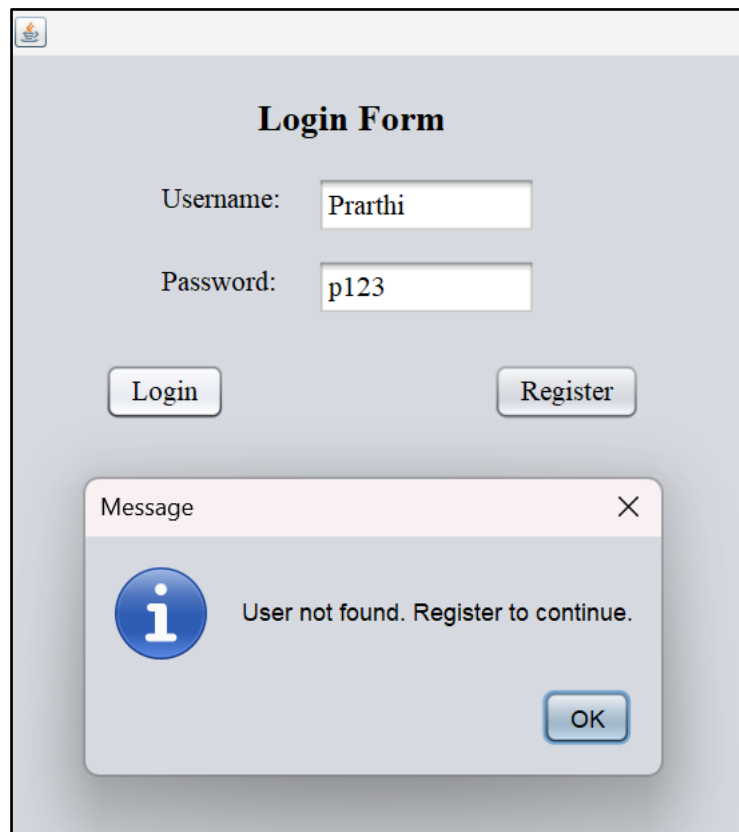
        String sql = "INSERT INTO REG_DETAILS VALUES('"+username+"','"+
            password+"')";
        smt.executeUpdate(sql);
        smt.close();
        con.close();
    }

    catch(SQLException se)
    {
        se.printStackTrace();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public static void main(String args[]) {
    LoginFormRegistration obj = new LoginFormRegistration ();
}
}

```

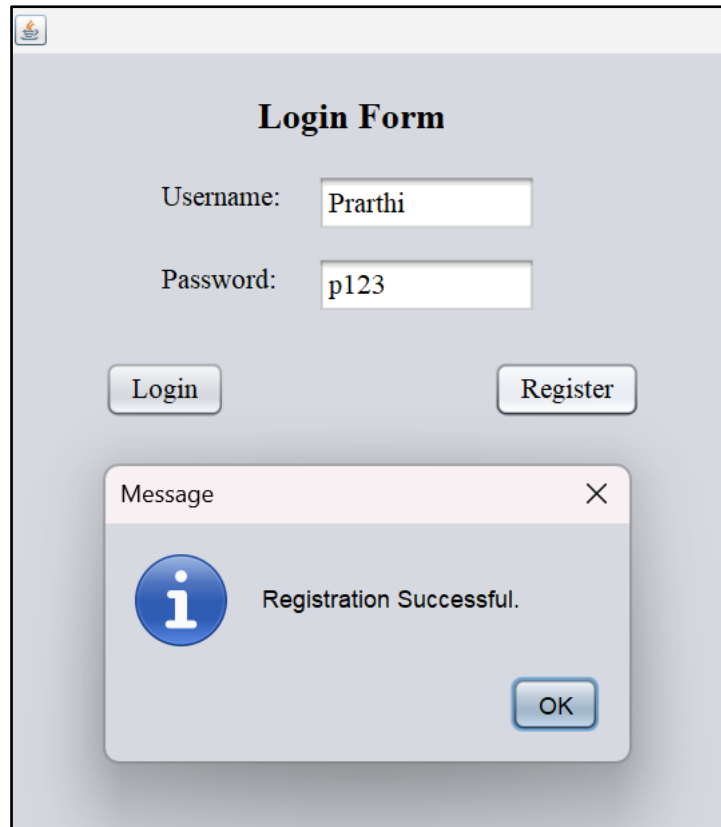
## OUTPUT –



**Figure 10:** SWING GUI for Login Form Registration – Output for new user

```
mysql> USE PRACT6;  
Database changed  
mysql> SELECT * FROM REG_DETAILS;  
Empty set (0.00 sec)  
  
mysql> |
```

**Figure 11:** Output for SQL – Initial empty set since user has not registered earlier

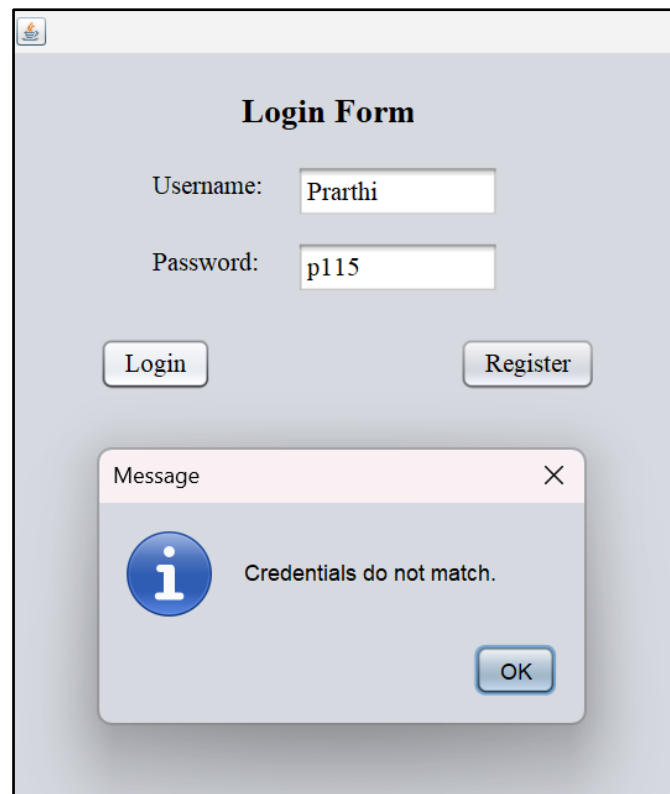


**Figure 12:** SWING GUI for Login Form Registration – Output for registration of new user

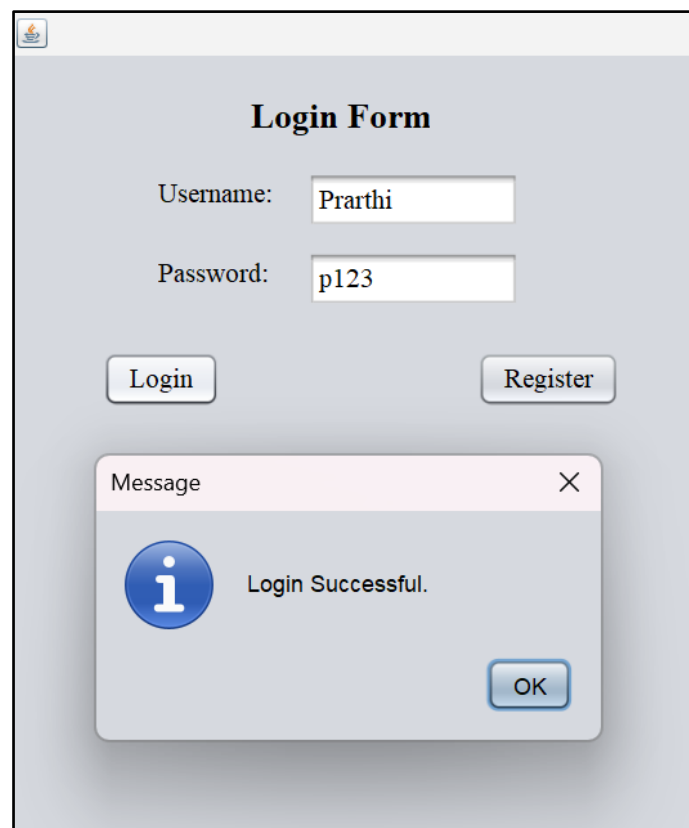
```
mysql> SELECT * FROM REG_DETAILS;
+-----+-----+
| USERNAME | PASSWORD |
+-----+-----+
| Prarthi  | p123     |
+-----+-----+
1 row in set (0.00 sec)

mysql> |
```

**Figure 13:** Output for SQL – User credentials successfully added to table REG\_DETAILS



**Figure 14:** SWING GUI for Login Form Registration – Output for entering incorrect credentials



**Figure 15:** SWING GUI for Login Form Registration – Output for entering correct credentials

**Question 3 –**

Create a JDBC for the following SWING Application GUI.

**CODE –**

```
package SWINGJDBC_Practs6;
import java.sql.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class StudentDetails extends JFrame implements ActionListener{

    JLabel jLabel1;
    JTextField jTextField1;
    JButton jButton1, jButton2;
    JTable jTable1;

    public StudentDetails() {
        jLabel1 = new JLabel("Enter Name:");
        jTextField1 = new JTextField(20);
        jButton1 = new JButton("SEARCH");
        jButton2 = new JButton("CLEAR DATA");
        jTable1 = new JTable();

        add(jLabel1);    add(jTextField1);
        add(jButton1);    add(jButton2);
        add(jTable1);

        jButton1.addActionListener(this);
        jButton2.addActionListener(this);

        setVisible(true);
        setSize(500,400);
    }

    private void jButton1ActionPerformed(ActionEvent evt) {
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Registered");
            Connection con;
            Statement smt;
            con=DriverManager.getConnection("jdbc:mysql://localhost/pract6", "root",
"root");
```

```

        System.out.println("Connection Successful");
        smt=con.createStatement();
        String sql="SELECT ID, NAME FROM STUDENT_DETAILS WHERE
NAME = " + jTextField1.getText()+" ";
        ResultSet rs=smt.executeQuery(sql);
        while(rs.next())
        {
            String id =String.valueOf(rs.getInt("ID"));
            String name=rs.getString("NAME");
            String tbldata[]={id,name};
            DefaultTableModel tbmodel=(DefaultTableModel)jTable1.getModel();
            tbmodel.addRow(tbldata);
        }
        smt.close();
        con.close();

        smt.close();
        con.close();
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

private void jButton2ActionPerformed(ActionEvent evt) {
    DefaultTableModel tbmodel=(DefaultTableModel)jTable1.getModel();
    tbmodel.setRowCount(0);
    jTextField1.setText("");
}

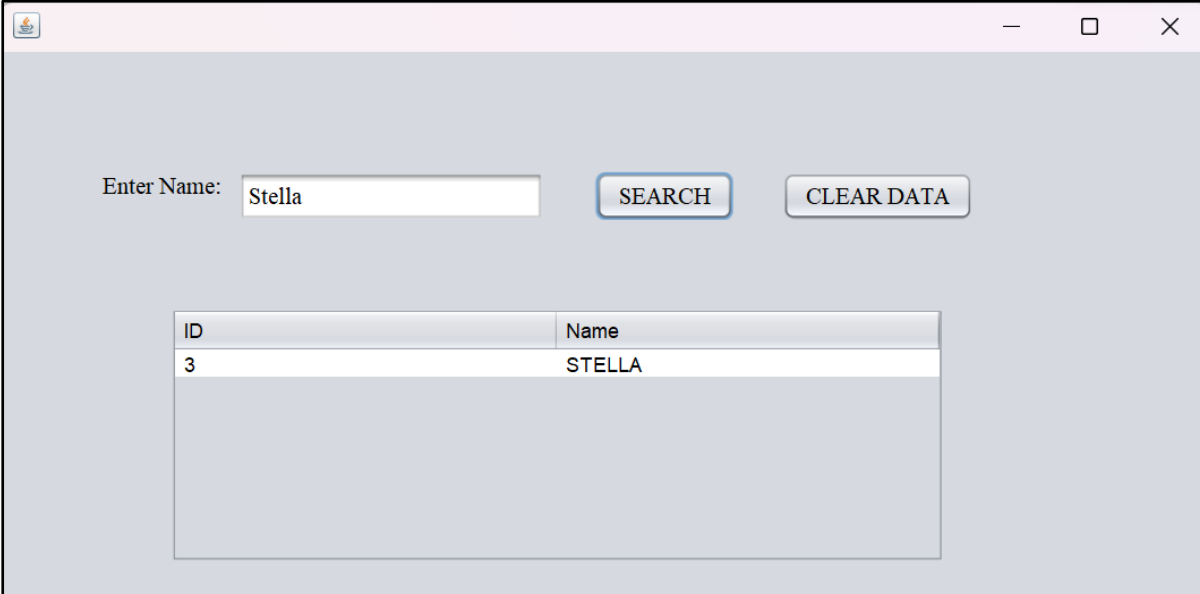
public static void main(String args[]) {
    StudentDetails obj = new StudentDetails();
}
}

```

## OUTPUT –

```
mysql> SELECT * FROM STUDENT_DETAILS;  
+-----+-----+  
| ID | NAME |  
+-----+-----+  
| 1 | ANDY |  
| 2 | ELIZABETH |  
| 3 | STELLA |  
+-----+-----+  
3 rows in set (0.02 sec)
```

**Figure 16:** Output for SQL – Information in STUDENT\_DETAILS table



Enter Name:

ID	Name
3	STELLA

**Figure 17:** SWING GUI for Student Details Search – Output for entering Name: “Stella”



Enter Name:

ID	Name
----	------

**Figure 18:** SWING GUI for Student Details Search – Output after clearing data

## **RESULTS:**

Using the Java programming language, the concept of Java Database Connectivity (JDBC) was explored and demonstrated in order to create GUI for accessing and editing SQL records using SQL queries and commands for registration of a new user, login form and retrieving details of a student by entering a name.

## **CONCLUSION:**

The concept of Java Database Connectivity (JDBC) in Java programming language was demonstrated.

## **REFERENCES:**

1. *JDBC Tutorial / What is Java Database Connectivity(JDBC)* - javatpoint. (n.d.).  
www.javatpoint.com. <https://www.javatpoint.com/java-jdbc>
  2. *Lesson: JDBC Introduction (The Java™ Tutorials > JDBC Database Access)*. (n.d.).  
<https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html>
  3. G. (2023, November 17). *Introduction to JDBC (Java Database Connectivity)*.  
GeeksforGeeks. <https://www.geeksforgeeks.org/introduction-to-jdbc/>
-



**Date – 17/02/2024**

## **Practical 7**

### **LINUX: DEMONSTRATION OF LINUX INSTALLMENT**

Perform the following:

1. Create a Linux partition. (Use either MS-DOS fdisk command or LINUX fdisk or diskdruid option).
2. Create boot disks for LINUX.
3. Install Red Hat Linux.
4. Understand the procedure of login, logout and shutting down the server.
5. Understanding the procedure for connecting to the Linux server and creating users either graphically or using Linuxconf utility.

### **Installation of Linux Red Hat 7.1**

We have 4.3 GB Hard disk drive installed in the system. 2GB is partitioned for Windows 98 as FAT 32. Rest disk space we will be using for Linux installation.

1. Enter into your system's CMOS setup, select Advanced BIOS setup (in most of the computers) and set the first boot drive as CDROM.
2. Insert the 1<sup>st</sup> CD into CDROM drive and start the computer system.
3. System will boot with the Linux Installation CD and will return to boot prompt.

#### **Boot:**

4. To install Linux in Graphical Mode. **Press <Enter>**

5. **Language Selection:**

Default is "English". Press <Enter>

6. **Keyboard Selection:**

Default is "US". Press <Enter>

7. **What type of system would you like to install ?**

**Available options are:**

- Workstation
- Server System
- Laptop
- Custom System
- Upgrade Existing Installation

**Select Custom System** as this allows you to select the partitions and keeps the existing partitions as they are all other types erase the existing partitions and we can loose the Windows 98 partition.

8. **Partitions:**

**Available options are:** Continually (Automatic)  
Manually Partition

Select “Manually Partition”.

9. **Tool to partition the Hard disk:**

**Available options are:** Disk Druid  
Fdisk

Select “Disk Druid”.

10. Now it will display the existing partition of windows 98 FAT32. The options are:

**Add / Edit / Delete / OK / Back**

Select “Add”. It will display the following box.

Mount Point:	Type : Linux Swap
Size in (Megs)	Linux Native
Use Remaining Space ? [ ]	DOS 16 bit < 32M
Allowable Drives [*] had	DOS 16 bit > 32M

Write:

**Mount Point:** /boot

**Size in (Megs):** 20

**Select Type:** Linux Native

Again select “Add” to create second partition.

**Mount Point:**

**Size in (Megs):** 64 (if you have 32MB RAM)

**Select Type:** Linux Swap

This will automatically write in Mount Point as “Swap Partition”

Again select “Add” to create third partition.

**Mount Point:** / (This ‘/’ will be root point)

**Size in (Megs):** 1000 (Considering only 2GB is available from 4GB HDD for Linux)

**Use Remaining Space [\*]** (Select ‘\*’ by pressing space bar)

**Select Type:** Linux Native

This will automatically write in Mount Point as “Swap Partition”

Now the main partition menu shows ‘0’ (zero) M free space since we have selected use remaining space in root partition.

Click ‘OK’. Save changes to your partition table ‘Yes’.

It will display the following message box:

We need to turn ON swap space immediately. To do this we will have to write your new partition table to the disk immediately. Is that Okay?

Select 'Yes'.

11. Choose partition to format by pressing spacebar. Select all the partitions we have created i.e., Boot and Root. Select 'OK'.

12.

LILO Configuration	
A few systems will need to pass special options to the Kernel at boot time for the system to function properly. If you need to pass boot options to the Kernel, enter them Now. If you don't need any or aren't sure , leave this blank.	

Leave this blank and press 'OK'.

13.

LILO Configuration	
/dev/had	Master Boot Record (MBR)
/dev/hda2	First sector of boot partition.

Select 'Master Boot Record' and press 'OK'.

14.

LILO Configuration	
The boot manager Red Hat uses can boot other operating systems as well. You need to tell me what partitions you would like to be able to boot and what level you want to use for each of them.	

Device	Partition Type	Default Boot Label
/dev/hda1	DOS / Windows	DOS
/dev/hda5	Linux Native	* Linux
OK      Edit      Back		

You can change the Boot Label if you want.

Select 'Edit' and change the label to windows instead of DOS.

'\*' is the first option you get while booting. (i.e. Linux will be the default operating system to load). You can change it to DOS / Windows by selecting that option and pressing Space Bar.

Finally Press 'OK'.

15. **Hostname Configuration:**

The hostname is the name of your computer. If your computer is attached to a network, this may be assigned by your network administrator.

Type the host name you want and select 'OK'.

16.

Firewall Configuration
A firewall protects against unauthorized network instructions. High security blocks all incoming accesses. Medium blocks access to system services (such as telnet or printing), but allows other connections. No firewall allows all connections and is not recommended.
Security Level : ( ) High ( * ) Medium ( ) No firewall
OK      Customize      Back

Select 'Medium' and press 'OK'.

17. **Mouse Selection:**

If you have serial mouse 3 button then:

Select **Generic – 3 Button Mouse (Serial)** from the list

[ ] Emulate 3 Buttons? - Leave this blank as it is.

Press 'OK'.

18.

Device
What device is your mouse located on ?
/dev/ttys0 (Com1 under DOS)
/dev/ttys1 (Com2 under DOS)
/dev/ttys2 (Com3 under DOS)
/dev/ttys3 (Com4 under DOS)

Select the highlighted one if you are not sure on which serial port your mouse is connected.

Press 'OK'.

19. **Language support:**

Select English (USA) by pressing spacebar on that.

[\*] English (USA).

Press 'OK'.

20. **Time Zone Selection:**

[ ] Hardware clock set to GMT?

Select from list 'Asia / Calcutta'.

Press 'OK'.

21. **Root Password:**

Enter Password.

Press 'OK'.

22.

Add User
You should use a normal user account for most activities on your system. By not using the root account casually, you will reduce the chance of disrupting your system's configuration.
User Id : _____
Password : _____
Password confirm : _____
Full Name : _____

Enter the above mentioned information and press 'OK'.

23.

Authentication Configuration
<input type="checkbox"/> * ] Use Shadow password
<input type="checkbox"/> * ] Enable MD5 passwords
<input type="checkbox"/> ] Enable NIS NIS Domain: _____
NIS server : <input type="checkbox"/> ] Request server via broad cast
Or use : _____
<input type="checkbox"/> ] Enable LDAP LDAP Server: _____
LDAP base DN:
<input type="checkbox"/> ] use TLS connections:
<input type="checkbox"/> ] Enable Kerberos Realm : _____
KDC : _____
Admin Server: _____

Select the default values and press 'OK'.

24.

Package Group Selection
<input type="checkbox"/> ]Printer support
<input type="checkbox"/> ]X windows system
<input type="checkbox"/> ]G Nome
<input type="checkbox"/> ]KDE
<input type="checkbox"/> ]Mail / www / News Tools
<input type="checkbox"/> ]DOS / windows connectivity
<input type="checkbox"/> ]Graphics Manipulation
<input type="checkbox"/> ]Games
<input type="checkbox"/> ]Multimedia support
<input type="checkbox"/> ]Laptop support
<input type="checkbox"/> ]Networked workstation
<input type="checkbox"/> ]Dialup workstation
<input type="checkbox"/> ]News server

[ ]NFS server
[ ]SMB (Samba) server
[ ]IPX / Netware <sup>TM</sup> connectivity
[ ]Anonymous FTP server
[ ]SQL server
[ ]Web server
[ ]DNS Name server
[ ]Network management workstation
[ ]Authoring / Publishing
[ ]Emacs
[ ]Development
[ ]Kernel development
[ ]Utilities
[ ]Everything

Select the options by pressing the space bar and finally press 'OK'.

25.

Video Card Selection
Select your video card from the list displayed in this box and then press 'OK'.

26.

Installation to Begin
A complete log of your installation will be in /tmp/install.log after rebooting your system. You may want to keep this file for later reference.

27. Formatting will start now followed by copying files. After completing this job, the installation procedure will ask to create boot disc.

28. Insert the blank floppy into floppy drive and press 'OK'.

29. **Monitor probe:**

Monitor probing found

Do you want to use these settings? - Press 'Yes'

30. **Video memory:** Select 1MB

31. **Clockchip configuration:** Select No clockchip settings (Recommended)

32. **Probe for clocks:** Select 'Probe'

33.

Select Video Modes		
8 Bit	16 Bit	24 Bit
<input type="checkbox"/> 1152 x 864 <input type="checkbox"/> 1024 x 768 <input type="checkbox"/> 800 x 600 <input type="checkbox"/> 640 x 480	<input type="checkbox"/> 800 x 600 <input checked="" type="checkbox"/> 640 x 480	<input type="checkbox"/> 640 x 480

Select the appropriate resolution by pressing spacebar and then press 'OK'.

34.

Starting X
X configuration will now start to test your configuration.

---

**Date – 19/02/2024**

## **Practical 8**

# **UNDERSTAND THE WORKING OF LINUX AND LINUX FILE SYSTEM**

1. Understand the working of LINUX (rules), LINUX file system.
2. Create your own user account from the root login. Enter login name, group, home directory and the password.
3. Understand the procedure of logging in and out of Linux.
4. Study the following user commands:
  - a. .passwd
  - b. .talk
  - c. .id
  - d. .pine
  - e. .su
  - f. .write
  - g. .users
  - h. .df
  - i. .who, who am i
  - j. .du
  - k. .clear
  - l. .login
5. Study of general purpose utilities: man, help, cal, banner, date, cal, history, tty, stty, echo, bc.
6. Study of directory commands: pwd, dir, cd, mkdir, rmdir

## **Answer 1**

### **Rules for naming and using files:**

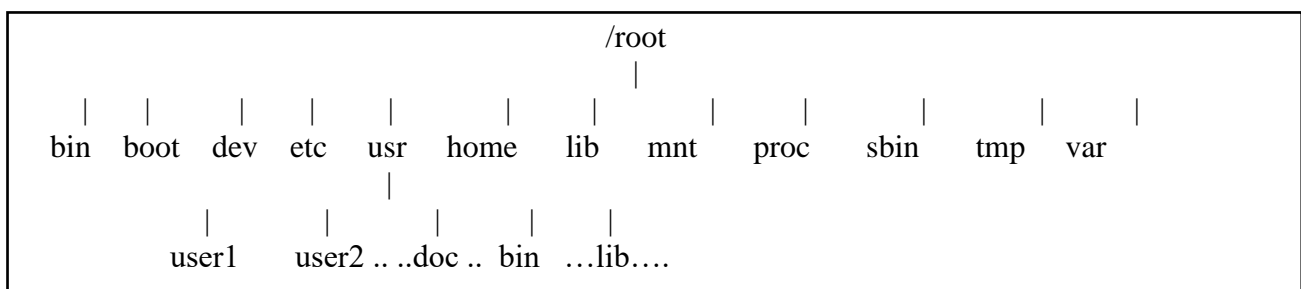
1. LINUX IS CASE-SENSITIVE. Your user login name and password are also case sensitive. (This goes with the tradition of UNIX and the "c" programming language being case sensitive.)
2. Filenames can be up to 256 characters long and can contain letters, numbers, "." (Dots), "\_" (underscores), "-" (dashes), plus some other non-recommended characters.
3. Files with names starting with "." are normally not shown by the ls (list) or dir command. Think of these "dot" files as "hidden". We use ls -a (list with the option "all") to see these files.
4. "/" is an equivalent to DOS "\" (root directory, meaning the parent of all other directories, or a separator between a directory name and a subdirectory or filename).



5. Under Linux, all directories appear under a single directory tree (there are no DOS-style drive letters). This means directories and files from all physical devices are merged into this single file system.
6. Linux is a multi-user system. Your personal settings are in your home directory which is `/home/your_user_login_name`.
7. Under Linux, as in any multi-user operating system, directories and files have an owner and set of permissions. You will typically be allowed to write only to your home directory which is `/home/your_user_login_name`. Learn to use the file permissions else you will be constantly annoyed with Linux.

### Linux file system:

Linux treats everything it knows and understands, as a file. All utilities, applications, data in Linux is stored as a file. Even a directory is treated as a file, which contains several other files. The file system, begins with a directory called root, and resembles an upside down tree. The root directory is denoted as (`/`). Different directories on the root directory are:



1. **/bin** – contains executable files for essential LINUX commands such as `cp`, `mv`, `rm`, `ln` etc. It also holds basic user programs, such as `login`, shells(`bash`, `tcsh`, `zsh` . . etc.)
2. **/boot** – contains files used during booting and possibly the kernel itself are stored here.
3. **/dev** - contains all files related to various devices connected to the system such as terminals, printer, disk drives etc.
4. **/etc** – contains the files required for system administration purpose such as networking, mail.. etc. It also contains the configuration files for the system, network and server. It contains subdirectories and files such as `fstab` (mounts file system when you start your system), `passwd`(contains user password and login configuration),.....etc.`shadow`(contains user's encrypted password).
5. **/usr** – contains user related programs and files. In `usr` directory there are several directories, each associated with a particular user. The system administrator creates these directories when he creates accounts for different users. Each user is allowed to work with his directory, often called as home directory. It can create subdirectories and files under his home directory.  
 /usr holds subdirectories such as `/bin`(holds programs for user's tasks), `/sbin` holds user-related system operation, such as user add to add users, `/lib` holds all the libraries used by the system.

6. **/home** - home directories of individual users. For each user its home directory is created with his user name when a user account is set up. /home also holds subdirectories such as /ftpd(for ftp files), /httpd(for Apache web server web site files).
7. **/lib** – contains all standard shared system library files/functions.
8. **/mnt** - typical mount point for many user-mountable devices such as floppy drives, cd-rom readers, etc. Each device is mounted on a subdirectory of /mnt.
9. **/proc** - virtual file system that provides a number of system statistics.
10. **/root** - home directory for root.
11. **/sbin** - location of binaries used for system administration, configuration, and monitoring.
12. **/tmp** - directory specifically designed for programs and users to store temporary files.
13. **/var** - administrative files such as log files, used by various utilities.

### **To connect through Telnet server**

Connect to another machine using the TELNET protocol. Use a remote machine name or IP address. You will be prompted for your login name and password--you must have an account on the remote machine to login. Telnet will connect you to another machine and let you operate on it as if you were sitting at its keyboard (almost). Telnet is not very secure--everything you type moves through the networks in open text, even your password!

### **Managing users**

Linuxconf: is a tool used for administrative tasks, including user and file management, as well as network services. It has three interfaces: text, GUI and HTML. The text interface provides cursor-based screens that can be run from shell command line. You use TAB key to move between boxes, lists and buttons. You use the arrow keys to select entries in a list. For GUI interface go to X window System interface that runs on any window manager or desktop. HTML interface is a web page interface that lists options as links to other web pages. You can download the current version from the Linuxconf Web site at [www.soulcorp.qc.ca/linuxconf](http://www.soulcorp.qc.ca/linuxconf).

You can easily add, remove, or change users with linuxconf:

1. Login as root user.
2. Use command linuxconf to go to the text interface.
3. Select user accounts in the normal list under the User accounts heading under Config. This displays a panel listing all your user accounts, including those used for special purposes.
4. To add a new user, click add on this panel. This displays a User information panel.
5. Enter the login name, group, and user's home directory. The root user assigns the permission at this stage, If not assigned, the minimum possible permissions will be assigned.
6. Click Accept. To give the user initial password, click Passwd. A changing Password panel is displayed where you can enter the new password.
7. Retype new UNIX password, click Accept. (It is possible to change the password). You now see the new user displayed in the user accounts panel. If you need to change or delete a user, double- click it entry in this panel to display the User information panel. To remove

the user, click Del. To edit the other entries, make changes at the appropriate entries and click Accept.

8. Quit enter 3 times. (It is not possible to add user if you log in as normal user. The user's entry is made in /home as well as in a file called passwd under /etc directory.)

### **Answer 2 and 3:**

As given in Practical 1

### **Answer 4:**

<i>User Commands for root user</i>	
<b>Command</b>	<b>Description and syntax</b>
Finger	Display information about a user # finger
Passwd	Change a user's password # passwd username
Su	Change to another user id # su username
Useradd	Add a new user to the system
Userdel	Delete a user from the system
Shutdown	To shut down or reboot the system # shutdown # shutdown -r now
Halt	halt the system
Init	set runlevel, or define processes that are begun on a specific runlevel # init 0
Lilo	install the LILO boot loader
Poweroff	power the system off
Reboot	reboot the system
Runlevel	show the current system runlevel

<i>User Commands for normal as well as root user</i>	
<b>Command</b>	<b>Description and syntax</b>
Passwd	Change a user's password # passwd username \$ passwd username password:
Id	Display information about a user # id \$ id
Su	Change to root user id \$ su rootuser/username password:

Users	Display a list of current users # users
who who am i	Display a list of current users as follows: Login name    terminal number    date &time Or the serial port Line by which your Terminal is connected To the host machine e.g. sup1   tty1   Dec 30 ..... # who   or \$ who # who am I or \$ who am i
Clear	Clears the screen \$ clear
Login	Initiate user login \$ login
Talk	Communicate between the two users: to talk
Pine	Communicate between the two users: to send the message
Write	Communicate between the two users: to send the message
Df	Display information about file system
Du	Display information about disk usage

<i>General Purpose Utilities</i>		
Command	Purpose	Syntax
Man	To get the on-line help. Offers help on almost all the topics related to linux.	\$man expr here expr denotes the command name for which you need the help
Help	To get the on-line help, but restricted to very few commands.	\$help expr here expr denotes the command name for which you need the help
Banner	To create a poster by blowing up its arguments on the keyboard. On each line it can display atleast ten characters.	\$banner expr OR \$/usr/games/banner expr expr denotes the expression which you display as the banner. \$banner -w n expr displays the banner of size n, where n is some positive integer.
date	To get the current date and time to the nearest second	\$date

	<p>For specific format each format is proceeded by a + symbol and the % operator</p> <p>To get the day</p> <p>To get the month in the number or name format.</p> <p>To get the year</p> <p>To get the date in the format mm/dd/yy</p> <p>To get the time in the format hr:min:sec</p> <p>To get the hours</p> <p>To get the minutes</p> <p>To get the seconds</p> <p><u>To couple more than one formats at a time enclose them in single or double quotes and write + symbol before</u></p>	<p>\$date +%d</p> <p>\$date +%m or +%h</p> <p>\$date +%y</p> <p>\$date +%D</p> <p>\$date +%T</p> <p>\$date +%D</p> <p>\$date +%M</p> <p>\$date +%S</p> <p>e.g</p> <p>\$date +%H:%M:%S"</p> <p>or</p> <p>\$date +%H:%M:%S'</p> <p><b>displays the time in the format hr:min:sec</b></p> <p>date +"%D%n%T"</p> <p>displays the date and and time on two different lines when %n is used between them.</p>
Cal	<p>To print the calendar of the current month of the current year</p> <p>To print the calendar for a particular year</p> <p>To print the calendar for a particular month of a year.</p>	<p>\$cal</p> <p>\$cal year</p> <p>\$cal month year</p> <p>where month and year denote the number of the month and year denotes the year.</p>
Tty	<p>To tell the name of the terminal(Linux treats even the terminal as file)</p>	<p>\$tty</p> <p>gives /dev/tty1</p> <p>which means the terminal name is tty1 in the /dev.</p> <p><b>(tty means the teletype command.)</b></p>
stty	<p>Used to set the terminal characteristics according to user's choice.</p>	<p>\$stty -a</p> <p>the output consists of the baud rate, the parameters , a series of keywords or options with a – preceded to some of them. When an option doesn't have the -, it means the option is turned on. It is possible to use stty to set or unset these options. Also it can be used to set the functions for some of the keys</p>

		e.g to change the interrupt key, to change the eof, eol character.
History	Every command has an event number associated with it. By default, Korn shell stores all previous commands. To see all the commands entered from the very beginning	\$history  \$history n prints last n number of commands typed.
Echo	To echo/display the expression required	\$echo expr displays the expr as it is.
Bc	Linux provides two calculators : bc and xcalc To do the mathematical calculations with truncation. To do the calculations with decimals.	\$bc or \$bc -q to get the clear bc \$bc -q scale = 2 to set the scale to get the truncation upto 2 decimal places while doing the calculations.
<i>Directory commands</i>		
pwd	When you are logged in, you are placed in a specific directory of the file system. This directory is called the current directory. To know the current working/home directory along with its pathname and parent directory.	\$pwd or \$echo \$HOME
Dir	To see the contents of the directory	\$dir <b>which displays all the files and directories</b> \$dir -a displays all the files and directories including the hidden files beginning with . and .. The symbols . and .. are used universally to represent the current and parent directory respectively.
Cd	To change to the parent directory	\$cd  \$cd dir1

	To change to some other directory say dir1	
Mkdir	To create a new directory with name dir_name To create the chain of directories	\$mkdir dir_name  \$mkdir -p dir1/dir2/dir3 creates dir3 inside dir2 inside dir1 on current working directory at once.
Rmdir	To remove already created directory with name dir_name To remove the chain of directories	\$rmdir dir_name  \$mkdir -p dir1/dir2/dir3 removes dir3, dir2 and dir1 at once.

---

**Date – 27/03/2024**

**Practical 9**  
**LINUX: BASIC LINUX COMMANDS**

**Part I : File System Commands: touch, cat, cp, rm, mv, mkdir, cd, rmdir.**

Do the following and enter the related commands in your journal.

1. Create five files with the name a1, b2, c3, d4, e5 by using touch command.
2. Create five files with the name f6, g7, h8 using cat command with some meaningful contents with at least five lines each.
3. Concatenate the contents of f6 and g7 to a file called new1. (Without creating new1)
4. Append the contents of f6 and h8 to new1.
5. Copy contents of f6 to a1, g7 to b2, h8 to c3 by using cp command.
6. Create two directories with the name dd1, dd2.
7. Copy the files a1 and b2 to the directory dd1 by using cp command. Copy the files f6, g7 to the directory dd2.
8. Remove the files a1, b2 from dd1.
9. Remove the directory dd2 along with its contents.
10. Rename the files f6, g7, h8, by newf6, newg7, newh8, using mv command.
11. Move the files newf6, newg7 to the directory dd1.

**Part II : ls and ls with options**

Change the directory to /bin and do the following and enter the commands in your journal.

1. List all filenames.
2. List all filenames with one screen at a time.
3. List all filenames with 2 characters, 3 characters.
4. List all filenames with 2 and 4 characters at the same time.
5. List all filenames starting with vowel.
6. List all filenames with the last character as a or b or c or d.
7. List all filenames with exactly three characters in which the second character is a vowel.
8. List all filenames starting with character 'a'.
9. List all 4 character filenames whose first character is 'a' and the third character is 'b'.
10. List all filenames whose first character is 'm' or 'r' or is in the range c to f or v to z.
11. List all filenames whose first character is any thing other than an alphabet in the range d to m.
12. Construct a command to display the total number of files with exactly three characters in their filename.
13. Construct a command to display the total number of files with exactly two or three or four characters in their filename.



Change the directory to your home directory and do the following:

1. List the contents of the directory.
2. List the contents of the directory along with all hidden files.
3. List the contents of the directory along with all hidden files except . and .. files.
4. List all files with their attributes and file permissions.
5. List all files identifying directories and executable files.
6. List all the files showing the size of each file rounded up to the nearest kilobyte.
7. List all the files according to file size.
8. Give the file listing displayed in columns.
9. Give the file listing in reverse order.
10. Give the file listing where all the files are in long format showing full file details.

### **Part-I:File System Commands:touch,cat,cp,rm,mv,mkdir,cd,rmdir**

1. **Create five files with the name a1,b2,c3,d4,e5 by using touch command.**

```
[tybsc308@linserver tybsc308]
```

```
$touch a1 b2 c3 d4 e5
```

2. **Create three files with the name f6,g7,h8 using cat command with some meaningful contents.**

```
[tybsc308@linserver tybsc308]$ cat> f6
```

```
This is file f6
```

```
this is cat command.
```

```
hello.
```

```
hi.
```

```
tybsc.
```

```
[tybsc308@linserver tybsc308]$ cat> g7
```

```
this is file g7
```

```
hello.
```

```
hi.
```

```
tybsc.
```

```
this is cat cmd.
```

```
[tybsc308@linserver tybsc308]$ cat> h8
```

```
this is file h8
```

```
hello.
```

```
hi.
```

```
tybsc.
```

```
this is cat cmd.
```

3. **Concatenate the contents of f6 and g7 to a file called new1.(Without creating new1)**

```
[tybsc308@linserver tybsc308]$ cat f6 g7 >new1
```

```
[tybsc308@linserver tybsc308]$ cat new1
```

```
This is file f6
```

```
this is cat command.
```

```
hello.  
hi.  
tybsc.  
thid is file g7  
hello.  
hi.  
tybsc.  
this is cat cmd.
```

**4. Append the contents of f6 and h8 to new1.**

```
[tybsc308@linserver tybsc308]$ cat f6 h8 >> new1  
[tybsc308@linserver tybsc308]$ cat new1  
This is file f6  
this is cat command.  
hello.  
hi.  
tybsc.  
thid is file g7  
hello.  
hi.  
tybsc.  
this is cat cmd.  
This is file f6  
this is cat command.  
hello.  
hi.  
tybsc.  
this is file h8  
hello.  
hi.  
tybsc.  
this is cat cmd.
```

**5. Copy contents of f6 to a1, g7 to b2, h8 to c3 by using cp command.**

```
[tybsc308@linserver tybsc308]$ cp f6 a1  
[tybsc308@linserver tybsc308]$ cat a1  
This is file f6  
this is cat command.  
hello.  
hi.  
tybsc.  
  
[tybsc308@linserver tybsc308]$ cp g7 b2  
[tybsc308@linserver tybsc308]$ cat b2  
thid is file g7  
hello.  
hi.  
tybsc.
```

this is cat cmd.

```
[tybsc308@linserver tybsc308]$ cp h8 c3
[tybsc308@linserver tybsc308]$ cat c3
this is file h8
hello.
hi.
tybsc.
this is cat cmd.
```

**6. Create two directory with the name dd1 and dd2.**

```
[tybsc308@linserver tybsc308]$ mkdir dd1
[tybsc308@linserver tybsc308]$ mkdir dd2
```

**7. Copy the files a1 and b2 to the directory dd1 by using cp command. Copy the files f6, g7 to the directory dd2.**

```
[tybsc308@linserver tybsc308]$ cp a1 b2 dd1
[tybsc308@linserver tybsc308]$ ls dd1
a1  b2
```

```
[tybsc308@linserver tybsc308]$ cp a1 b2 dd2
[tybsc308@linserver tybsc308]$ ls dd1
a1  b2
```

**8. Remove the files a1, b2 from dd1.**

```
[tybsc308@linserver tybsc308]$ rm dd1/a1
[tybsc308@linserver tybsc308]$ ls dd1
b2
```

```
[tybsc308@linserver tybsc308]$ rm dd1/b2
[tybsc308@linserver tybsc308]$ ls dd1
```

**9. Remove the directory dd2 along with its contents.**

```
[tybsc308@linserver tybsc308]$ rm -r dd2
```

**10. Rename the files f6, g7, h8, by newf6, newg7, newh8, using mv command.**

```
[tybsc308@linserver tybsc308]$ mv f6 newf6
[tybsc308@linserver tybsc308]$ mv g7 newg7
[tybsc308@linserver tybsc308]$ mv h8 newh8
```

**11. Move the files newf6, newg7 to the directory dd1.**

```
[tybsc308@linserver tybsc308]$ mv newf6 newg7 dd1
[tybsc308@linserver tybsc308]$ ls dd1
newf6  newg7
```

## **Part II: ls and ls with options**

**Change the directory to /bin and do the following and enter the commands in your journal.**

### **1. List all filenames.**

```
[tybsc308@linserver tybsc308]$ cd /bin
[tybsc308@linserver bin]$ ls
arch          cut          gawk         ls           red
tcsh
ash           date        gettext      mail         rm
touch
ash.static    dd          grep         mkdir        rmdir
true
aumix-minimal df          gtar         mknod        rpm
umount
awk           dmesg       gunzip       mktemp       rvi
uname
basename     dnsdomainname gzip         more         rview
unicode_start
bash          doexec      hostname     mount        sed
unicode_stop
bash2         domainname  igawk        mt           setfont
unlink
bsh           dumpkeys   ipcalc       mv
setserial    usleep
cat           echo       jpeg-6b      netstat      sh
vi
chgrp         ed          kbd_mode    nice         sleep
view
chmod         egrep       kill         nisdomainname sort
ypdomainname
chown         env         link         pgawk        stty
zcat
cp            ex          ln           ping         su
cpio          false      loadkeys    ps           sync
csh           fgrep      login        pwd          tar
```

### **2. List all filenames with one screen at a time.**

```
[tybsc308@linserver bin]$ ls |more
arch
ash
ash.static
aumix-minimal
awk
basename
bash
bash2
bsh
cat
chgrp
```

```
chmod
chown
cp
cpio
csh
cut
date
dd
df
dmesg
dnsdomainname
doexec
--More--
```

**3. List all filenames with 2 characters, 3 characters.**

```
[tybsc308@linserver bin]$ ls ??
cp dd df ed ex ln ls mt mv ps rm sh su vi

[tybsc308@linserver bin]$ ls ???
ash awk bsh cat csh cut env pwd red rpm rvi sed
tar
```

**4. List all filenames with 2 characters and 4 characters at the same time.**

```
[tybsc308@linserver bin]$ ls ?? ????
arch cpio df ex gtar link mail mv ps sort sync
vi bash date echo gawk gzip ln more nice rm stty
tcsh view cp dd ed grep kill ls mt ping sh
su true zcat
```

**5. List all filenames starting with vowel**

```
[tybsc308@linserver bin]$ ls [aeiou]*
arch ash.static awk ed env igawk umount
unicode_start unlink
ash aumix-minimal echo egrep ex ipcalc uname
unicode_stop usleep
```

**6. List all filenames with the last character as a or b or c or d.**

```
[tybsc308@linserver tybsc308]$ ls *[abcd]
fspaa fspac newsed prime.c xaa xac
fspab masterdata primel.c transdata xab
```

**7. List all filenames with exactly three characters in which the second character is a vowel.**

```
[tybsc308@linserver bin]$ ls ?[aeiou]?
cat cut red sed tar
```

**8. List all filenames starting with character 'a'.**

```
[tybsc308@linserver bin]$ ls [a]*
```

```
arch ash ash.static aumix-minimal awk
```

- 9. List all 4 character filenames whose first character is 'a' and the third character is 'b'.**

```
[tybsc308@linserver bin]$ ls [a]?[b]?  
ls: [a]?[b]?: No such file or directory
```

- 10. List all filenames whose first character is 'm' or 'r' or is in the range c to f or v to z.**

```
[tybsc308@linserver bin]$ ls [mrc-fv-z]*  
cat      cpio  df      dumpkeys  ex      mknod   mv      rvi  
zcat  
chgrp   csh    dmesg    echo      false   mktemp  red  
rview  
chmod   cut    dnsdomainname  ed      fgrep   more    rm      vi  
chown   date   doexec    egrep    mail    mount   rmdir   view  
cp      dd     domainname  env      mkdir   mt      rpm  
ypdomainname
```

- 11. List all filenames whose first character is any thing other than an alphabet in the range d to m.**

```
[tybsc308@linserver bin]$ ls [!d-m]*  
arch      cat      nice      rpm      stty  
unicode_start  
ash      chgrp    nisdomainname  rvi      su  
unicode_stop  
ash.static  chmod    pgawk      rview    sync    unlink  
aumix-minimal  chown    ping      sed      tar      usleep  
awk      cp      ps      setfont  tcsh    vi  
basename  cpio     pwd      setserial touch    view  
bash      csh      red      sh      true  
ypdomainname  
bash2     cut      rm      sleep    umount  zcat  
bsh      netstat  rmdir    sort     uname
```

- 12. Construct a command to display the total number of files with exactly three characters in their filename.**

```
[tybsc308@linserver bin]$ ls ???|wc -l  
13
```

- 13. Construct a command to display the total number of files with exactly two or three or four characters in their filename.**

```
[tybsc308@linserver bin]$ ls ?? ??? ????|wc -l  
49
```

**Change the directory to the home directory and do the following**

- 14. List the contents of the directory.**

```
[tybsc308@linserver tybsc308]$ ls dir1  
f1 file1
```

- 15. List the contents of the directory along with all hidden files.**

```
[tybsc308@linserver tybsc308]$ ls -a dir1
.  ..  f1  file1
```

**16. List the contents of the directory along with all hidden files except . and .. files.**

```
[tybsc308@linserver tybsc308]$ ls -A
+1          copy1          fcmp1_308    green        pract7.10
pract7.7    prime1.out    Student     date         fcmp2
.gtkrc      pract7.11    pract7.9    prime.c      Student308
ashwini     date1        fcmp2_308   .kde         pract7.12
pract8.1    .prime.c.swo  t4          .bash_history dir1
fcmp3_308   line         pract7.13   pract8.11.2  prime.out
transdata   .bash_logout dir2         fex1         login
pract7.15   pract8.12.3  product_308 try2
.bash_profile dir3          .file       loop         pract7.17
pract8.14   result       tryTime     .bashrc      doll
file2       masterdata   pract7.18   pract8.3     result1
.viminfo    bdata308     .emacs      fodl         merit308
pract7.3    pract8.5     Shweta308   yellow
check       emp          foreg       month        pract7.3.1
pract8.6    ssl          checkgrade  empdata308   fsp308
names       pract7.4     pract8.7    ss2          commandfor
fact        gre1         pract       pract7.5     pract8.8
ss3         commandline  fcmp1
```

**17. List all files with their attributes and file permissions.**

```
[tybsc308@linserver tybsc308]$ ls -l
total 204
-rw-r--r--  1 tybsc308 tybsc          159 Dec  7 10:52 +1
-rw-r--r--  1 tybsc308 tybsc          16 Jan 18 09:57
ashwini
-rw-r--r--  1 tybsc308 tybsc        193 Jan 18 08:39 check
-rw-r--r--  1 tybsc308 tybsc        307 Jan 10 15:51
Stud308
...
...
-rw-r--r--  1 tybsc308 tybsc        139 Jan 19 11:53
Student
-rw-r--r--  1 tybsc308 tybsc        881 Jan  4 08:58 t1
-rw-rw-r--  1 tybsc308 546          881 Dec 14 09:52 try1
-rw-rw-r--  1 tybsc308 546           28 Dec 30 10:15 white
-rw-rw-r--  1 tybsc308 546           36 Dec 14 09:09 yellow
```

**18. Same as above**

**19. List all the files showing the size of each file rounded up to the nearest kilobyte.**

```
[tybsc308@linserver tybsc308]$ ls -s
total 204
  4 +1                4 dir3                4 login                4 pract7.17
4 ss1
  4 ashwini           4 doll                4 loop                4 pract7.18
4 ss2
```

4 check	4 emp	4 names	4 pract7.3
4 ss3			
4 checkgrade	4 fact	4 pink	4 pract7.3.1
4 Stud308			
4 commandfor	4 file2	4 pract	4 pract7.4
4 Student			
4 date	4 gre1	4 pract7.11	0 pract7.7
4 white			
...			

## 20. List all the files according to file size.

```
[tybsc308@linserver tybsc308]$ ls -S
```

prime.out	prime.c	month	fact	fcmp3_308
bdata308	fcmp1_308	pract7.18	prime1.out	pract8.14
pract7.11	+1	loop	merit308	commandline
pract7.3	dir1	pract7.13	pract7.12	pract8.12.3
Student308	transdata	fod1	10000	
dir2	pract8.5	pract7.10	pract8.8	pract7.17
file2	doll	date1	dir3	prime1.c
checkgrade	Student	line	result	pract7.1
pract7.7	result1	tryTime	pract7.9	pract7.4
ss3	pract8.1	yellow	fsp308	pract7.3.1
date	Shweta308	masterdata	emp	green
login	pract8.6	pract7.15	pract8.7	names
fcmp2_308	commandfor	t4	pract8.11.2	empdata308
pract7.6	pract7.5	fcmp2	ss2	try2
gre2	copy1	gre1	foreg	product_308
pract8.3	fex1	Stud308	check	pract
fcmp1	ss1	ashwini		

## 21. Give the file listing displayed in columns.

```
[tybsc308@linserver tybsc308]$ ls -c
```

gre2	date1	pract7.18	check	ss1	
pract7.1	white	gre1	commandfor	ashwini	emp
	checkgrade	fsp308	green	+1	
pract8.3	foreg	ss3	pract7.13	pract7.7	Stud308
ss2	dir2	dir3			
Student	pract7.15	pract7.12	doll	names	t1
login	dir1	pract7.11	pract7.17	fact	
pract7.6	copy1	commandline	try1	pract8.1	
pract	pract7.10	pract7.4	pract7.3.1	file2	
yellow	date	loop	pract7.9	pract7.5	
pract7.3	pink	line			

## 22. Give the file listing in reverse order.

```
[tybsc308@linserver tybsc308]$ ls -r
```

yellow	ss2	pract7.5	pract7.13	names	
fsp308	dir2	white	checkgrade	ss1	
pract7.4	pract7.12	loop	foreg	dir1	check
try1	pract8.3	pract7.3.1	login	file2	date1
ashwini	t1	pract8.1	line	pract7.3	



pract7.10	fact	date	+1	Student
pract7.9	pract7.18	pract7.1	green	
emp	copy1	Stud308	pract7.7	pract7.17
pract				
gre2	doll	commandline	ss3	pract7.6
pract7.15				
pink	gre1	dir3	commandfor	

### 23. Give the file listing where all the files are in long format showing full size details.

```
[tybsc308@linserver tybsc308]$ ls -l
```

```
total 204
-rw-r--r-- 1 tybsc308 tybsc 159 Dec 7 10:52 +1
-rw-r--r-- 1 tybsc308 tybsc 16 Jan 18 09:57 ashwini
-rw-r--r-- 1 tybsc308 tybsc 193 Jan 18 08:39 check
-rw-r--r-- 1 tybsc308 tybsc 243 Jan 18 08:38 checkgrade
-rw-r--r-- 1 tybsc308 tybsc 31 Jan 18 11:26 commandfor
-rw-r--r-- 1 tybsc308 tybsc 56 Jan 4 08:57
commandline
-rw-r--r-- 1 tybsc308 tybsc 212 Jan 7 11:56 copy1
-rw-r--r-- 1 tybsc308 tybsc 220 Jan 18 11:48 date
-rw-r--r-- 1 tybsc308 tybsc 0 Jan 18 11:32 datel
drwxrwxr-x 2 tybsc308 546 4096 Nov 30 09:44 dir1
drwxrwxr-x 3 tybsc308 546 4096 Dec 14 10:08 dir2
drwxrwxrwx 2 tybsc308 546 4096 Dec 7 09:07 dir3
-r-xr-xr-x 1 tybsc308 tybsc 47 Jan 4 08:24 doll
. . .
-rw-r--r-- 1 tybsc308 tybsc 41 Jan 4 10:18 pract7.1
-rw-r--r-- 1 tybsc308 tybsc 260 Jan 18 09:10 pract7.10
-rw-r--r-- 1 tybsc308 tybsc 299 Jan 18 09:24 pract7.11
-rw-r--r-- 1 tybsc308 tybsc 286 Jan 18 09:33 pract7.12
-rw-r--r-- 1 tybsc308 tybsc 419 Jan 18 09:56 pract7.13
-rw-r--r-- 1 tybsc308 tybsc 220 Jan 18 11:03 pract7.15
-rw-r--r-- 1 tybsc308 tybsc 115 Jan 18 10:49 pract7.17
-rw-r--r-- 1 tybsc308 tybsc 3 Jan 18 10:26 pract7.18
-rw-r--r-- 1 tybsc308 tybsc 1 Jan 4 10:32 pract7.3
-rw-r--r-- 1 tybsc308 tybsc 360 Jan 4 10:49 pract7.3.1
-rw-r--r-- 1 tybsc308 tybsc 138 Jan 18 08:10 pract7.4
-rw-r--r-- 1 tybsc308 tybsc 103 Jan 18 08:02 pract7.5
-rw-r--r-- 1 tybsc308 tybsc 128 Jan 18 08:12 pract7.6
-rw-r--r-- 1 tybsc308 tybsc 0 Jan 18 08:33 pract7.7
-rw-r--r-- 1 tybsc308 tybsc 221 Jan 18 08:43 pract7.9
-rw-r--r-- 1 tybsc308 tybsc 79 Jan 19 11:18 pract8.1
-rw-rw-r-- 1 tybsc308 546 28 Dec 30 10:15 white
-rw-rw-r-- 1 tybsc308 546 36 Dec 14 09:09 yellow
```

## **Practical 10**

### **LINUX: STUDY OF COMMANDS: find, tr, head, tail, wc, file, sort, split**

**1. Display using “find command” all the filenames under:**

- (i) /usr/sbin one screen at a time
- (ii) /usr/sbin beginning with a lowercase ‘c’.
- (iii) /usr/sbin in capital letters beginning with a lowercase ‘c’  
(Use single-quote for tr command)
- (iv) /usr/sbin which are over 5k in size in uppercase.

**2. Display Parts of Files using head or tail command:**

- (i) Display and count all the lines in the file /etc/mime.types
- (ii) Display the first 10 lines of the file /etc/mime.types
- (iii) Display the last 10 lines of /etc/mime.types
- (iv) Display the first 25 lines of /etc/mime.types

**3. Classify, Count and Compare Files**

a. Find out what file types you have in the following directories:

- (i) /etc
- (ii) /bin

b. Repeat the previous question, but this time:

- (i) Re-direct /etc listing to new file etcfiles.txt
- (ii) Append the listing for /usr/bin to etcfiles.txt

c. Construct a command to find out how many files are in the /usr/bin directory.

**4. Sorting:**

- (i) Sort the etcfiles.txt file into reverse alphabetical order on the first field. You may notice that capital and lowercase letters are sorted independently, e.g. ‘A’ comes before ‘a’.
- (ii) Repeat the first sorting exercise but ignoring case differences.
- (iii) Sort the etcfiles.txt files into alphabetical order on the second field (the file type).
- (iv) Find out how many English text files are listed in the etcfiles.txt file.

**5. Create a file using vi editor with the following contents**

Mahesh Deshpande 234  
Naresh Nair 431  
Allen Disuza 121  
Hari Kutian 231  
Ramesh Dubey 231  
Akshay Das 256

- (i) Sort on the first names only
- (ii) Sort on last names only
- (iii) Sort on first 4 characters only
- (iv) Sort on their numbers only.

**6. Construct and execute the commands to create a file with the name Stud<roll\_no> with the following fields separated by a blank space having the below mentioned values:**

Field	RollNo	First Name	Last Name	Date of Birth	Marks
Values	Numeric	Character	Character	dd-mm-yy	Numeric out of 600

Insert at least five appropriate records and do the following:

- a. Sort the data on first names only.
- b. Sort the data on the Marks only.
- c. Prepare a ranked merit list with student's first and last name only and store in the file Merit<roll\_no> and display its contents.

**Write down the commands and attach the printout of the commands and their the corresponding output in your answer sheet.**

**7. Construct the commands and execute them to:**

- (i) Create a file named fsp<seat\_no> having the listing of      atleast 50 lines (e.g, listing of /usr/sbin or /usr/bin or /etc or can create your own).
- (ii) Display first 2 lines of fsp<seat\_no> and convert all the characters into capital letters.
- (iii) Display the last 15 lines of fsp<seat\_no>.
- (iv) Display the lines starting with a vowel.
- (v) Split the file fsp<seat\_no> into subparts each having at most 20 lines and display the contents of these subparts and count the number of lines in them.
- (vi) Split the file fsp<seat\_no> into three subparts named fspaa, fspab, fspac and display the contents of these files and count the number of lines in them.

**1. Display using “find command” all the filenames under:**

**(i) /usr/sbin one screen at a time.**

```
[tybsc308@linserver tybsc308]$ find /usr/sbin | more
/usr/sbin
/usr/sbin/iconvconfig
/usr/sbin/rpcinfo
/usr/sbin/build-locale-archive
/usr/sbin/zdump
/usr/sbin/zic
/usr/sbin/pwunconv
/usr/sbin/pwck
/usr/sbin/glibc_post_upgrade
/usr/sbin/alternatives
/usr/sbin/update-alternatives
/usr/sbin/mklost+found
/usr/sbin/arping
/usr/sbin/clockdiff
/usr/sbin/ping6
/usr/sbin/rdisc
/usr/sbin/tracepath
/usr/sbin/tracepath6
/usr/sbin/traceroute6
/usr/sbin/adduser
/usr/sbin/chpasswd
/usr/sbin/groupadd
/usr/sbin/groupdel
--More--
```

**(ii) /usr/sbin beginning with a lowercase 'c'.**

```
[tybsc308@linserver tybsc308]$ find /usr/sbin/c*
/usr/sbin/camel-index-control
/usr/sbin/camel-lock-helper
/usr/sbin/capiinit
/usr/sbin/chat
/usr/sbin/chkfontpath
/usr/sbin/chpasswd
/usr/sbin/chroot
/usr/sbin/ciped-cb
/usr/sbin/clockdiff
/usr/sbin/crond
/usr/sbin/cupsaddsmb
/usr/sbin/cupsd
```

**(iii) /usr/sbin in capital letters beginning with a lowercase 'c'. (Use single-quote for tr command)**

```
[tybsc308@linserver tybsc308]$ find /usr/sbin/c* | tr '[a-z]'
'[A-Z]'
/USR/SBIN/CAMEL-INDEX-CONTROL
/USR/SBIN/CAMEL-LOCK-HELPER
/USR/SBIN/CAPIINIT
/USR/SBIN/CHAT
/USR/SBIN/CHKFONTPATH
/USR/SBIN/CHPASSWD
/USR/SBIN/CHROOT
/USR/SBIN/CIPED-CB
/USR/SBIN/CLOCKDIFF
/USR/SBIN/CROND
/USR/SBIN/CUPSADDSMB
/USR/SBIN/CUPSD
```

**(iv) /usr/sbin which are over 5k in size in uppercase.**

```
[tybsc308@linserver tybsc308]$ find /usr/sbin -size 5k | tr '[a-
z]' '[A-Z]'
/USR/SBIN/MODELINE2FB
/USR/SBIN/RPC.NFSD
/USR/SBIN/EXECCAP
/USR/SBIN/SETPCAPS
```

## **2. Display parts of files using head or tail command:**

**(i) Display and count all the lines in the file/etc/mime.types**

```
[tybsc308@linserver tybsc308]$ wc -l /etc/mime.types
480 /etc/mime.types
```

**(ii) Display the first 10 lines of the file/etc/mime.types**

```
[tybsc308@linserver tybsc308]$ head -10 /etc/mime.types
# This is a comment. I love comments.
```

```
# This file controls what Internet media types are sent to the
client for
# given file extension(s). Sending the correct media type to
the client
# is important so they know how to handle the content of the
file.
# Extra types can either be added here or by using an AddType
directive
# in your config files. For more information about Internet
media types,
# please read RFC 2045, 2046, 2047, 2048, and 2077. The
Internet media type
# registry is at <http://www.iana.org/assignments/media-
types/>.
```

**(iii) Display the last 10 lines of the /etc/mime.types**

```
[tybsc308@linserver tybsc308]$ tail -10 /etc/mime.types
```

```
video/vnd.fvt
video/vnd.motorola.video
video/vnd.motorola.videop
video/vnd.mpegurl          mxu
video/vnd.mts
video/vnd.nokia.interleaved-multimedia
video/vnd.vivo
video/x-msvideo            avi
video/x-sgi-movie          movie
x-conference/x-cooltalk    ice
```

**(iv) Display the first 25 lines of the /etc/mime.types**

```
[tybsc308@linserver tybsc308]$ head -25 /etc/mime.types
```

```
# This is a comment. I love comments.
```

```
# This file controls what Internet media types are sent to the
client for
```

```
# given file extension(s). Sending the correct media type to
the client
```

```
# is important so they know how to handle the content of the
file.
```

```
# Extra types can either be added here or by using an AddType
directive
```

```
# in your config files. For more information about Internet
media types,
```

```
# please read RFC 2045, 2046, 2047, 2048, and 2077. The
Internet media type
```

```
# registry is at <http://www.iana.org/assignments/media-
types/>.
```

```
# MIME type                                Extension
```

```
application/EDI-Consent
```

```
application/EDI-X12
```

```
application/EDIFACT
```

```
application/activemessage
```

```
application/andrew-inset          ez
```

```
application/applefile
```

```
application/atomicmail
```

```
application/batch-SMTP
```

```
application/beep+xml
```

```
application/cals-1840
```

```
application/commonground
```

```
application/cybercash
```

```
application/dca-rft
```

```
application/dec-dx
```

### 3. Classify, Count and Compare Files

a) Find out what file types you have in the following directories:

(i) /etc

```
[tybsc308@linserver tybsc308]$ file /etc/*
/etc/a2ps.cfg: ASCII English text
/etc/a2ps-site.cfg: ASCII English text
/etc/adjtime: ASCII text
/etc/aep: directory
/etc/aep.conf: ASCII text
/etc/aeplog.conf: ASCII text
/etc/alchemy: directory
/etc/aliases: ASCII English text
/etc/aliases.db: can't read `/etc/aliases.db'
(Permission denied).
/etc/alternatives: directory
/etc/anacrontab: ASCII text
/etc/at.deny: can't read `/etc/at.deny'
(Permission denied).
/etc/auto.master: ASCII English text
/etc/auto.misc: ASCII English text
/etc/bashrc: ASCII text
/etc/bonobo-activation: directory
/etc/cdrecord.conf: ASCII English text
/etc/cipe: directory
/etc/CORBA: directory
/etc/cron.d: directory
/etc/cron.daily: directory
/etc/cron.hourly: directory
/etc/cron.monthly: directory
/etc/crontab: ASCII text
/etc/cron.weekly: directory
/etc/csh.cshrc: ASCII text
/etc/csh.login: ASCII text
/etc/cups: directory
/etc/default: directory
/etc/DIR_COLORS: ASCII English text
/etc/DIR_COLORS.xterm: ASCII English text
....
....
```

(ii) /bin

```
[tybsc308@linserver tybsc308]$ file /usr/bin/*
/usr/bin/zip: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
```

```

/usr/bin/zipcloak: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zipgrep: Bourne shell
script text execu
table
/usr/bin/zipinfo: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zipnote: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zipsplit: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zless: Bourne shell
script text execu
table
/usr/bin/zmore: Bourne shell
script text execu
table
....

```

**b) Repeat the previous question, but this time:**

**(i) Re-direct /etc listing to new file etcfiles.txt**

```

[tybsc308@linserver tybsc308]$ file /etc/*>etcfiles.txt
[tybsc308@linserver tybsc308]$ cat etcfiles.txt | more
/etc/a2ps.cfg: ASCII English text
/etc/a2ps-site.cfg: ASCII English text
/etc/adjtime: ASCII text
/etc/aep: directory
/etc/aep.conf: ASCII text
/etc/aeplog.conf: ASCII text
/etc/alchemy: directory
/etc/aliases: ASCII English text
/etc/aliases.db: can't read `/etc/aliases.db'
(Permission denied).
/etc/alternatives: directory
/etc/anacrontab: ASCII text

```



```

/etc/at.deny:                can't read `/etc/at.deny'
(Permission denied).
/etc/auto.master:            ASCII English text
/etc/auto.misc:              ASCII English text
/etc/bashrc:                 ASCII text
/etc/bonobo-activation:      directory
/etc/cdrecord.conf:          ASCII English text
/etc/cipe:                   directory
/etc/CORBA:                  directory
/etc/cron.d:                 directory
/etc/cron.daily:             directory
/etc/cron.hourly:            directory
/etc/cron.monthly:           directory

```

**(ii) Append the listing for /usr/bin to etcfiles.txt**

```

[tybsc308@linserver tybsc308]
$ file /usr/bin/*>> etcfiles.txt

```

**c) Construct a command to find out how many files are in the usr/bin directory.**

```

[tybsc308@linserver tybsc308]$ file /usr/bin/* | wc -l
2161

```

## 4. Sorting

**a) Sort the etcfiles.txt into reverse alphabetical order on the first file.**

```

[tybsc308@linserver tybsc308]$ sort -r etcfiles.txt | more
/usr/bin/zsoelim:                symbolic
link to soelim
/usr/bin/zsoelim:                symbolic
link to soelim
/usr/bin/znew:                   Bourne shell
ELF 32-bit LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zipnote:                ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped

```

**b) Repeat the first sorting exercise but ignoring case differences.**

```

[tybsc308@linserver tybsc308]$ sort -f -r etcfiles.txt | more
/usr/bin/zsoelim:                symbolic
link to soelim
/usr/bin/zsoelim:                symbolic
link to soelim

```

```

/usr/bin/znew: Bourne shell
script text execu
table
/usr/bin/znew: Bourne shell
script text execu
table
/usr/bin/zmore: Bourne shell
script text execu
table
/usr/bin/zmore: Bourne shell
script text execu
table
/usr/bin/zless: Bourne shell
script text execu
table
/usr/bin/zless: Bourne shell
script text execu
table
/usr/bin/zipsplit: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zipsplit: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped
/usr/bin/zipnote: ELF 32-bit
LSB executable, Int
el 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared
libs), stripped

```

**c) Sort the etcfiles.txt into alphabetical order on the second field**

```

[tybsc3
08@linserver tybsc308]$ sort +1 -2 etcfiles.txt | more
/usr/bin/evolution-move-tasks: a perl
script text executable
/usr/bin/evolution-move-tasks: a perl
script text executable
/usr/bin/libglade-xgettext: a python
script text executabl
e
/usr/bin/libglade-xgettext: a python
script text executabl
e
/etc/rpc: ASCII C program text

```

```

/etc/mime-magic:          ASCII C++ program text
/etc/ltrace.conf:        ASCII C program text
/usr/bin/e2pall:          ASCII English text
/usr/bin/e2pall:          ASCII English text
/usr/bin/xsubpp:          ASCII English text
/usr/bin/xsubpp:          ASCII English text
/usr/bin/epstopdf:        ASCII English text
/usr/bin/epstopdf:        ASCII English text
/usr/bin/thumbpdf:        ASCII English text
/usr/bin/thumbpdf:        ASCII English text
/etc/imrc:                ASCII English text
/etc/fdprm:               ASCII English text
/etc/hosts:               ASCII English text

```

**d) Find out how many English text files are listed in the etcfiles.txt file**

```

[tybsc308@linserver tybsc308]$ grep English etcfiles.txt | wc -l
67

```

**5. Create a file using vi editor with the following contents.**

```

[tybsc308@linserver tybsc308]$ cat names
Mahesh Deshpande 234
Naresh Nair 431
Allen Disuza 121
Hari Kutian 231
Ramesh Dubey 231
Akshay Das 256

```

**(i) Sort on the first names only**

```

[tybsc308@linserver tybsc308]$ sort +0 -1 names
Akshay Das 256
Allen Disuza 121
Hari Kutian 231
Mahesh Deshpande 234
Naresh Nair 431
Ramesh Dubey 231

```

**(ii) Sort on the last names only**

```

[tybsc308@linserver tybsc308]$ sort +1 -2 names
Akshay Das 256
Mahesh Deshpande 234
Allen Disuza 121
Ramesh Dubey 231
Hari Kutian 231
Naresh Nair 431

```

**(iii) Sort on first four characters only**

```
[tybsc308@linserver tybsc308]$ cut -c 1-4 names | sort
Aksh
Alle
Hari
Mahe
Nare
Rame
```

**(iv) Sort on their numbers only**

```
[tybsc308@linserver tybsc308]$ sort +2 -3 names
Allen Disuza 121
Hari Kutian 231
Ramesh Dubey 231
Mahesh Deshpande 234
Akshay Das 256
Naresh Nair 431
```

**6. Construct and execute the commands to create a file with the name Stud<roll\_no> with the following fields separated by a blank space.**

**(i) Insert at least five appropriate records and do the following:**

```
[tybsc308@linserver tybsc308]$ cat Stud308
308      Shweta      Choudhary    7-12-1985    450
309      Mildred     D'mello      9-12-1985    500
310      Megha       Menon        14-10-1985    550
311      Vikrant     Mahkal       12-5-1985    500
312      Sankalp     Naik         11-5-1985    550
```

**(ii) Sort the data on first names only**

```
[tybsc308@linserver tybsc308]$ sort +1 -2 Stud308
RollNo  FirstName  LastName  DateOfBirth  Marks
308     Megha    Menon     14-10-1985    550
308     Mildred  D'mello   9-12-1985    500
308     Sankalp  Naik      11-5-1985    550
308     Shweta   Choudhary 7-12-1985    450
308     Vikrant  Mahkal    12-5-1985    500
```

**(iii) Sort the data on the Marks only**

```
[tybsc308@linserver tybsc308]$ sort +4 -5 Stud308
308     Shweta      Choudhary    7-12-1985    450
308     Mildred     D'mello      9-12-1985    500
308     Vikrant     Mahkal       12-5-1985    500
308     Megha       Menon        14-10-1985    550
308     Sankalp     Naik         11-5-1985    550
RollNo  FirstName  LastName  DateOfBirth  Marks
```

## 7. Construct the commands and execute them to:

### (i) Create a file named fsp<seat\_no> having of atleast 50 lines

```
[tybsc308@linserver tybsc308]$ touch fsp308
[tybsc308@linserver tybsc308]$ head -50 /etc/mime.types >
fsp308
[tybsc308@linserver tybsc308]$ cat fsp308
# This is a comment. I love comments.

# This file controls what Internet media types are sent to the
client for
# given file extension(s).  Sending the correct media type to
the client
# is important so they know how to handle the content of the
file.
# Extra types can either be added here or by using an AddType
directive
# in your config files.  For more information about Internet
media types,
# please read RFC 2045, 2046, 2047, 2048, and 2077.  The
Internet media type
# registry is at <http://www.iana.org/assignments/media-
types/>.
# MIME type                                Extension
application/EDI-Consent
...
application/news-message-id
application/news-transmission
application/ocsp-request
application/ocsp-response
```

### (ii) Display first two lines of fsp<seat\_no> and convert all the characters into capital letters

```
[tybsc308@linserver tybsc308]$ head -2 fsp308 | tr '[a-z]'
'[A-Z]'
# THIS IS A COMMENT. I LOVE COMMENTS.
```

### (iii) Display the last 15 lines of fsp<seat\_no>

```
[tybsc308@linserver tybsc308]$ tail -15 fsp308
application/iotp
application/ipp
application/isup
application/font-tdpfr
application/mac-binhex40          hqx
application/mac-compactpro        cpt
application/macwriteii
application/marc
application/mathematica
```

```

application/mathematica-old
application/msword          doc
application/news-message-id
application/news-transmission
application/ocsp-request
application/ocsp-response

```

**(iv) Display the lines starting with a vowel**

```

[tybsc308@linserver tybsc308]$ grep ^[aeiouAEIOU] fsp308
application/EDI-Consent
application/EDI-X12
application/EDIFACT
application/activemessage
application/andrew-inset    ez
application/applefile
application/atomicmail
application/batch-SMTP
application/beep+xml
application/cals-1840
application/commonground
application/cybercash
application/dca-rft
application/dec-dx
application/dvcs
application/eshop
application/http
application/hyperstudio
application/iges
application/index
application/index.cmd
. . .
application/mathematica-old
application/msword          doc
application/news-message-id
application/news-transmission
application/ocsp-request
application/ocsp-response

```

**(v) Split a file fsp<seat\_no> into subparts each having at most 20 lines and display the contents of these subparts and count the number of lines in them**

```

[tybsc308@linserver tybsc308]$ split -20 fsp308
[tybsc308@linserver tybsc308]$ ls
+1          date1          fcmp3_308    line          t4
10000       dir1          fcut1        login         ashwini
dir2        fcut2        loop         product_308  try2
bdata308    dir3         fex1         masterdata   result
tryTime     check        doll         file2        merit308

```

result1	xaa	emp	fod1	month
pract7.18	Shweta308	xab	empdata308	foreg
names	pract7.3	ss1	xac	commandline
fact	fsed1	newsed	ss2	yellow
copy1	fcmp1	fsp308	newsedclear	ss3
cutlist1	fcmp1_308	gre1	pract	
cutlist2	fcmp2	gre2	pract7.1	pract7.6
primel.c				
date	fcmp2_308	green	pract7.10	pract7.7
primel.out	Student308			

```
[tybsc308@linserver tybsc308]$ cat xaa | wc -l
20
[tybsc308@linserver tybsc308]$ cat xab | wc -l
20
[tybsc308@linserver tybsc308]$ cat xac | wc -l
10
```

- (vi) Split the files fsp<seat\_no> into three subparts named fspaa, fspab, fspac and display the contents of these files and count the number of lines in them

```
[tybsc308@linserver tybsc308]$ split -20 fsp308 fsp
[tybsc308@linserver tybsc308]$ ls
+1          date1          fcmp3_308    line          t4
10000       dir1          fcut1        login         ashwini
dir2        fcut2        loop         product_308  try2
bdata308    dir3         fex1         masterdata    result
tryTime     check        doll         file2         merit308
result1     xaa          emp          fod1          month
pract7.18   Shweta308    xab          empdata308    foreg
names       pract7.3     fspaa        xac           commandline
fact        fsed1        newsed       fspab         yellow
copy1       fcmp1        fsp308       newsedclear   fspac
cutlist1    fcmp1_308   gre1         pract
cutlist2    fcmp2        gre2         pract7.1      pract7.6
primel.c
date        fcmp2_308   green        pract7.10     pract7.7
primel.out  Student308
```

```
[tybsc308@linserver tybsc308]$ cat fspaa | wc -l
20
[tybsc308@linserver tybsc308]$ cat fspab | wc -l
20
[tybsc308@linserver tybsc308]$ cat fspac | wc -l
10
```

**Practical 11**  
**LINUX: COMPARING FILES**

**1. od, cmp, comm, diff, uniq:**

**Create a file named fod1 with some content shaving the following contents and display it in**

**(i) octal form only and (ii) octal form along with its text contents.**

**2. Construct the commands to**

- a. Create a file fcmp1<seat\_no> with six lines containing six names.
- b. Add two more names and save the contents in fcmp2<seat\_no>.
- c. Sort the contents of file fcmp1<seat\_no> .
- d. Display the names, which are common to fcmp1 <seat\_no>, and fcmp2<seat\_no>.
- e. Display the difference between fcmp1 <seat\_no> and fcmp2<seat\_no>.
- f. Append the contents of fcmp1 <seat\_no> to fcmp2<seat\_no> and store it in the file fcmp3<seat\_no>.
- g. Sort the contents of fcmp3<seat\_no> and display the contents without any duplicate lines.

**3. Create two files named fcmp1 and fcmp2 which consists of at least five lines with two or three similar lines. Construct the commands**

**A. using cmp**

- (i) to check whether the files differ
- (ii) to compare the two files byte by byte.

**B. using diff**

- (i) to display the lines which are common to both , the lines which are not common and to display the lines, which are common to both
- (ii) to display the difference in context output format
- (iii) to display the unified output format.

**C. Using comm**

- (i) To compare the files fcmp1 and fcmp2
- (ii) To display the lines which are unique to fcmp1 and fcmp2
- (iii) To display the lines which are common to the fcmp1 and fcmp2.

**D. Using nl to give the line numbers to lines in fcmp1.**

**E. Using cp to append the fcmp1 to fcmp2 and sort this appended fcmp2 and store it in a file named funiq1.**



F. Using uniq

- (i) to remove the duplicate lines in funiq1.
- (ii) to count the duplications and prepend number to each line
- (iii) to display the duplicate lines only
- (iv) to display unique lines only.

**Write down the commands and attach the printout of the commands and their the corresponding output in your answer sheet.**

**4. Create a file named name<roll\_no> with fields (first name, second name, last name, salary) separated by “ : ”. Insert at least five appropriate records in above file.**

**Perform following sort operations:**

- (i) Sort on first names only.
- (ii) Display only those records whose first names start with a vowel.
- (iii) Sort on last names only.
- (iv) Display the names with salary above 10000 and add two more records and redirect the output to the file named namenew<roll\_no>.

**Write the commands in the answer sheet, execute and attach the printout of the commands with their output.**

**5. Cut, paste, tr**

Create two files with at least three fields(columns) each with the names fcut1, fcut2 and do the following:

- (i) Cut first two columns from fcut1 and store the contents in the file cutlist1 and cut the second and the third column from the fcut2 and store it in cutlist2.
- (ii) Paste the contents of cutlist2 to contents of cutlist1.

Translate the first three lines into capital letters using tr command.

## 1. od,cmp,comm.,diff,uniq:

Create a file named fod1 with some contents having the following contents and display it in:

### (i) octal form only

```
[tybsc308@linserver tybsc308]$ od -b fod1
0000000 124 150 151 163 040 151 163 040 160 162 141 143 164
151 143 141
0000020 154 040 156 165 155 142 145 162 040 065 012 123 150
167 145 164
0000040 141 040 103 150 157 165 144 150 141 162 171 012 124
131 102 163
0000060 143 012
0000062
```

### (ii) octal form along with its text contents.

```
[tybsc308@linserver tybsc308]$ od -b -c fod1
0000000 124 150 151 163 040 151 163 040 160 162 141 143 164
151 143 141
          T   h   i   s           i   s           p   r   a   c   t
i   c   a
0000020 154 040 156 165 155 142 145 162 040 065 012 123 150
167 145 164
          l           n   u   m   b   e   r           5   \n   S   h
w   e   t
0000040 141 040 103 150 157 165 144 150 141 162 171 012 124
131 102 163
          a           C   h   o   u   d   h   a   r   y   \n   T
Y   B   s
0000060 143 012
          c   \n
0000062
```

## 2. Construct the commands to:

### a. Create a file fcmp1<seat\_no> with six lines containing six names.

```
[tybsc308@linserver tybsc308]$ vi fcmp1_308
[tybsc308@linserver tybsc308]$ cat fcmp1_308
Shweta
Megha
Mildred
Sonal
Nikita
Nisha
```

- b. Add two more names and save the contents in fcmp2<seat\_no>.**  
[tybsc308@linserver tybsc308]\$ cp fcmp1\_308 fcmp2\_308 && cat  
>> fcmp2\_308  
Minal  
Ankit  
[tybsc308@linserver tybsc308]\$ cat fcmp2\_308  
**Shweta**  
**Megha**  
**Mildred**  
**Sonal**  
**Nikita**  
**Nisha**  
**Minal**  
**Ankit**
- c. Sort the contents of file fcmp1<seat\_no>.**  
[tybsc308@linserver tybsc308]\$ sort fcmp1\_308  
Ankit  
Megha  
Mildred  
Minal  
Nikita  
Nisha  
Shweta  
Sonal
- d. Display the names, which are common to fcmp1<seat\_no>, and fcmp2<seat\_no>.**  
[tybsc308@linserver tybsc308]\$ comm fcmp1\_308 fcmp2\_28  
Shweta  
Megha  
Mildred  
Sonal  
Nikita  
Nisha  
Minal  
Ankit
- e. Display the difference between fcmp1<seat\_no> and fcmp2<seat\_no>.**  
[tybsc308@linserver tybsc308]\$ cat >> fcmp1\_308  
**Abhishek**  
[tybsc308@linserver tybsc308]\$ diff fcmp1\_308 fcmp2\_308  
9d8  
< Abhishek

- f. **Append the contents of fcmp1<seat\_no> to fcmp2<seat\_no> and store it in the file fcmp3<seat\_no>.**

```
[tybsc308@linserver tybsc308]$ cat fcmp1_308 fcmp2_308
>fcmp3_308
[tybsc308@linserver tybsc308]$ cat fcmp3_308
Shweta
Megha
Mildred
Sonal
Nikita
Nisha
Minal
Ankit
Shweta
Megha
Mildred
Sonal
Nikita
Nisha
Minal
Ankit
Abhishek
```

- g. **Sort the contents of fcmp3<seat\_no> and display the contents without and duplicate lines.**

```
[tybsc308@linserver tybsc308]$ sort -u fcmp3_308
Abhishek
Ankit
Megha
Mildred
Minal
Nikita
Nisha
Shweta
Sonal
```

3. **Create two files named fcmp1 and fcmp2 which consists of at least five lines with two or three similar lines. Construct the commands.**

```
[tybsc308@linserver tybsc308]$ cat fcmp1
This is fcmp1
This is fcmp2
This is unix practical
Practical on od
Practical no 5

[tybsc308@linserver tybsc308]$ cat fcmp2
This is fcmp1
This is fcmp2
```

```
This is 3rd line
Goodbye
Practical no 5
```

#### **A)Using cmp**

##### **(i) To check whether the files differ.**

```
[tybsc308@linserver tybsc308]$ cmp fcmp1 fcmp2
fcmp1 fcmp2 differ: byte 37, line 3
```

##### **(ii) To compare the two files byte to byte.**

```
[tybsc308@linserver tybsc308]$ cmp -l fcmp1 fcmp2
37 165 63
38 156 162
39 151 144
40 170 40
41 40 154
42 160 151
43 162 156
44 141 145
45 143 12
. . .
59 141 151
60 154 143
61 40 141
62 157 154
63 156 40
64 40 156
66 144 40
67 12 65
68 120 12
cmp: EOF on fcmp2
```

#### **B) Using diff**

##### **(i) To display the lines which are common to both, the lines which are not common and to display the lines, which are common to both.**

```
[tybsc308@linserver tybsc308]$ diff fcmp1 fcmp2
3,4c3,4
< This is unix practical
< Practical on od
---
> This is 3rd line
> Goodbye
```

##### **(ii) To display the difference in context output format.**

```
[tybsc308@linserver tybsc308]$ diff -c fcmp1 fcmp2
*** fcmp1      2006-01-24 09:30:21.000000000 +0530
--- fcmp2      2006-01-24 09:35:02.000000000 +0530
```

```

*****
*** 1,5 ****
    This is fcmp1
    This is fcmp2
! This is unix practical
! Practical on od
    Practical no 5
--- 1,5 ----
    This is fcmp1
    This is fcmp2
! This is 3rd line
! Goodbye
    Practical no 5

```

**(iii) To display the unified output format.**

```

[tybsc308@linserver tybsc308]$ diff -u fcmp1 fcmp2
--- fcmp1      2006-01-24 09:30:21.000000000 +0530
+++ fcmp2      2006-01-24 09:35:02.000000000 +0530
@@ -1,5 +1,5 @@
    This is fcmp1
    This is fcmp2
-This is unix practical
-Practical on od
+This is 3rd line
+Goodbye
Practical no 5

```

**C) Using comm**

**(i) To compare the files fcmp1 and fcmp2**

```

[tybsc308@linserver tybsc308]$ comm fcmp1 fcmp2
This is fcmp1
This is fcmp2
This is 3rd line
Goodbye
Practical no 5
This is unix practical
Practical on od
Practical no 5

```

**(ii) To display the lines which are unique to fcmp1 and fcmp2**

```

[tybsc308@linserver tybsc308]$ comm -3 fcmp1 fcmp2
Goodbye
Practical no 5
This is unix practical
Practical on od
Practical no 5

```

**(iii) To display the lines which are common to fcmp1 and fcmp2**

```
[tybsc308@linserver tybsc308]$ comm -12 fcmp1 fcmp2
This is fcmp1
This is fcmp2
This is 3rd line
```

**D) Using nl to give the line numbers to lines in fcmp1.**

```
[tybsc308@linserver tybsc308]$ nl fcmp1
1 This is fcmp1
2 This is fcmp2
3 This is unix practical
4 Practical on od
5 Practical no 5
```

**E) Using cat to append the fcmp1 to fcmp2 and sort this appended fcmp2 and store it in a file named funiq1**

```
[tybsc297@linserver tybsc297]$ cat fcmp1
Hello
Hi
GoodMorning
GoodNight
Good Bye
```

```
[tybsc297@linserver tybsc297]$ cat fcmp2
Hello
Hi
GoodMorning
GoodNight
Good Bye
```

```
Hello
Hi
GoodMorning
GoodNight
Good Bye
```

```
[tybsc297@linserver tybsc297]$
cat fcmp1 fcmp2 |sort > funiq1
[tybsc297@linserver tybsc297]$ cat funiq1
Good Bye
Good Bye
Good Bye
GoodMorning
GoodMorning
GoodMorning
GoodNight
GoodNight
GoodNight
```

```
Hello
Hello
Hello
Hi
Hi
Hi
```

**F) Using uniq**

**(i) To remove the duplicate lines in funiq1**

```
[tybsc297@linserver tybsc297]$ uniq funiq1
Good Bye
GoodMorning
GoodNight
Hello
Hi
```

**(ii) To count the duplications and prepend number to each line**

```
[tybsc297@linserver tybsc297]$ uniq -c funiq1
3 Good Bye
3 GoodMorning
3 GoodNight
3 Hello
3 Hi
```

**(iii) To display the duplicate lines only**

```
[tybsc297@linserver tybsc297]$ uniq -d funiq1
Good Bye
GoodMorning
GoodNight
Hello
Hi
```

**(iv) To display unique lines only**

```
[tybsc297@linserver tybsc297]$ uniq funiq1
Good Bye
GoodMorning
GoodNight
Hello
Hi
```



- 4. Create a file name<roll\_no> with fields(first name, second name, last name, salary)separated by “:”. Insert at least five appropriate records in above file.**

```
[tybsc308@linserver tybsc308]$ vi Shweta308
[tybsc308@linserver tybsc308]$ cat Shweta308
Shweta:Naresh:Choudhary:20000
Anu:Chandu:Menon:30800
Nisha:Suresh:Choudhary:2500
Mildred:Felix:D'mello:40000
Dilip:Chandu:Mehra:60000
```

- (i) Sort on first names only.**

```
[tybsc308@linserver tybsc308]$ sort +1 -2 Shweta308
Anu:Chandu:Menon:30800
Dilip:Chandu:Mehra:60000
Mildred:Felix:D'mello:40000
Nisha:Suresh:Choudhary:2500
Shweta:Naresh:Choudhary:20000
```

- (ii) Display only those records whose first names start with a vowel.**

```
[tybsc308@linserver tybsc308]$ grep ^[AEIOU] Shweta308
Anu:Chandu:Menon:30800
```

- (iii) Sort on last names only.**

```
[tybsc308@linserver tybsc308]$ sort -t ":" +2 -3 Shweta308
Shweta:Naresh:Choudhary:20000
Nisha:Suresh:Choudhary:2500
Mildred:Felix:D'mello:40000
Dilip:Chandu:Mehra:60000
Anu:Chandu:Menon:30800
```

- (iv) Display the names with salary above 10000 and add two more records and redirect the output to the file namednew<roll\_no>.**

```
[tybsc308@linserver tybsc308]$ grep '[1-9][0-9][0-9][0-9][0-9]' Shweta308
Shweta:Naresh:Choudhary:20000
Anu:Chandu:Menon:30800
Mildred:Felix:D'mello:40000
Dilip:Chandu:Mehra:60000
```

## 5. Cut,paste,tr

**Create two files with at least three fields(columns) each with the names fcut1, fcut2 and do the following:**

```
[tybsc308@linserver tybsc308]$ cat fcut1
Akshay Das 256
Allen Disuza 121
Hari Kutian 231
Mahesh Deshpande 234
Naresh Nair 431
```

```
[tybsc308@linserver tybsc308]$ cat fcut2
This is fcml
This is fcml2
This is 3rd line
Goodbye
Practical no 5
```

**(i) Cut first two columns from fcut1 and store the contents in the file cutlist1 and cut the second and third columns from the fcut2 and store it in cutlist2.**

```
[tybsc308@linserver tybsc308]$ cut -c 1-2 fcut1 > cutlist1
[tybsc308@linserver tybsc308]$ cat cutlist1
Ak
Al
Ha
Ma
Na
```

```
[tybsc308@linserver tybsc308]$ cut -c 2-3 fcut2 > cutlist2
[tybsc308@linserver tybsc308]$ cat cutlist2
hi
hi
hi
oo
ra
```

**(ii) Paste the contents of cutlist2 to contents of cutlist1.**

```
[tybsc308@linserver tybsc308]$ paste cutlist2 cutlist1
[tybsc308@linserver tybsc308]$ cat cutlist1
Ak
Al
Ha
Ma
Na
hi
hi
hi
oo
ra
```

**Translate the first three lines into capital letters using tr command.**

```
[tybsc308@linserver tybsc308]$ head -3 fcut1 fcut2 |
```

```
tr '[a-z]' '[A-Z]'
```

```
==> FCUT1 <==
```

```
AKSHAY DAS 256
```

```
ALLEN DISUZA 121
```

```
HARI KUTIAN 231
```

```
==> FCUT2 <==
```

```
THIS IS FCMP1
```

```
THIS IS FCMP2
```

```
THIS IS 3RD LINE
```

---

**Date – 01/03/2024**

**Practical 12**  
**LINUX: Grep, egrep, fgrep**

A) Create the file with the name gre1 and the following contents:

This is a first line.  
This is a second line.  
Please type the third line.  
Do you wish to continue?  
Simply type the fourth line.

B) Create the file with the name gre2 and the following contents:

The grep is an acronym for ‘globally search a regular expression and print it’. The command searches the specified input globally for a match with the specified pattern and displays it. While forming the pattern to be searched we can use shell metacharacters, or regular expressions as professional unix users call them.

C) Do the following:

1. Search for the word ‘line’ and display the lines containing it.
2. Search for the word ‘the or ‘The’ in both the files gre1 and gre2 and display the lines containing it.
3. Search for 4 letter words in gre1 and gre2 whose first character is ‘r’ and last character is ‘r’ .
4. Display the lines, which end with the characters from s to z from gre1 and gre2.

D) Create a file empdata<seatno>, which contains following fields.

Fieldname	Datatype	Value
Employee last name	character	
Employee first name	character	
Employee code	numeric	Starts with letter ‘E’
Permanent address	character	
Department code	character	MKT, HRD, PUR
Grade	character	A-C
Years of experience	numeric	
Date of birth	dd-mm-yy	
Basic pay	numeric	

Insert at least *five* records in above file.

Character fields in each record may not be in the same case. '~' is used as a field separator.

Give commands to

1. Display all employees who are not in department MKT. Display the output sorted on department code and grade.
2. Display all employees whose years of service are more than 5.
3. Store employee name and date of birth in a file bdata<seatno>.
4. Count total number of employees whose department code is HRD.

Write down the commands in the answer sheet, execute them and show them to the examiner.

E) Create a file student<seatno> with following fields

Field Name	Datatype	Values
Student code	character	
<i>Student name</i>	<i>character</i>	
Batch code	character	Q11 - Q15
No. of modules	Numeric	1 - 5
Average marks	Numeric	

Fields are separated by ":" (colon). Insert at least *five* appropriate records and give the commands to

1. Display the details of student in order of their name ignoring case.
2. Display the details of student whose number of modules is greater than 3.
3. Store the list of first 5 rank holders in merit<seatno> file.
4. Count number of students in Batch Q13.
5. Display the list of students with same names.

Write down the commands in the answer sheet, execute them and show them to the examiner.

**A) Create the file with the name gre1 and the following contents:**

```
[tybsc308@linserver tybsc308]$ cat gre1
```

**This is a first line.**

**This is a second line.**

**Please type the third line.**

**Do you wish to continue?**

**Simply type the fourth line.**

**B) Create the file with the name gre2 and the following contents:**

```
[tybsc308@linserver tybsc308]$ cat gre2
```

**The grep is an acronym for 'globally search a regular expression and print it'.**

**The command searches the specified input globally for a match with the specified pattern and displays it. While forming the pattern to be searched we can use shell metacharacters, or regular expressions as professional unix users call them.**

**C) Do the following:**

**1) Search the word 'line' and display the lines containing it.**

```
[tybsc308@linserver tybsc308]$ grep line gre1
```

**This is a first line.**

**This is a second line.**

**Please type the third line.**

**Simply type the fourth line.**

**2) Search the word 'the or The' in both the files gre1 and gre2 and display the lines containing it.**

```
[tybsc308@linserver tybsc308]$ grep the gre1 gre2
```

**gre1:Please type the third line.**

**gre1:Simplet type the fourth line.**

**gre2:The command searches the specified input globally for a match with the specified pattern and displays it. While forming the pattern to be searched we can use shell metacharacters, or regular expressions as professional unix users call them.**

```
[tybsc308@linserver tybsc308]$ grep The gre1 gre2
```

**gre2:The grep is an acronym for 'globally search a regular expression and print it'.**

**gre2:The command searches the specified input globally for a match with the specified pattern and displays it.**

**While forming the pattern to be searched we can use shell metacharacters, or regular expressions as professional unix users call them.**

- 3) Search for 4 letters in gre1 and gre2 whose first character is 'r' and last character is 'r'**

```
[tybsc308@linserver tybsc308]$ grep r.....r gre1 gre2
gre2:The command searches the specified inpyt globally
for a match with the specified pattern and displays it.
While forming the pattern to be searched we can use shell
metacharacters, or regular expressions as professional
unix users call them.
```

- 4) Display the lines, which end with the characters from s to z from gre1 and gre2**

```
[tybsc308@linserver tybsc308]$ grep '[s-z]$_' gre1 gre2
gre1:This is a first line from a to z
gre2:The grep is an acronym for globally search a regular
expression and print it
```

- D) Create a file empdata<seatno>, which contains the following fields.**

```
[tybsc308@linserver tybsc308]$ cat empdata308
Menon~Reena~E01~Mumbai~MKT~A~1~07-12-74~10000
Rao~Leena~E02~Madras~HRD~B~2~08-09-78~1000
Nair~Anu~E03~Dadar~PUR~A~6~10-12-85~50000
Shah~Meena~E04~Mahim~HRD~B~5~16-11-85~2500
Naik~Rupa~E05~Bandra~MKT~A~1~15-05-89~30800
```

- (i) Display all employees who are not in department MKT. Display the output sorted on department code and grade.**

```
[tybsc308@linserver tybsc308]$ grep -v MKT empdata308 | sort
+5 -7
Nair~Anu~E03~Dadar~PUR~A~6~10-12-85~50000
Rao~Leena~E02~Madras~HRD~B~2~08-09-78~1000
Shah~Meena~E04~Mahim~HRD~B~5~16-11-85~2500
```

- (ii) Display the employees whose years of service are more than 5.**

```
[tybsc308@linserver tybsc308]$ grep '~[5-9]*~' empdata308
Nair~Anu~E03~Dadar~PUR~A~6~10-12-85~50000
Shah~Meena~E04~Mahim~HRD~B~5~16-11-85~2500
```

- (iii) Display employee(s) who get highest salary**

- (iv) Store employee name and date of birth in a file bdata<seatno>.**

```
[tybsc308@linserver tybsc308]$ cut -f 1,2,8 -d"~" empdata308
> bdata308
[tybsc308@linserver tybsc308]$ cat bdata308
Menon~Reena~07-12-74
Rao~Leena~08-09-78
Nair~Anu~10-12-85
Shah~Meena~16-11-85
Naik~Rupa~15-05-89
```

**(v) Count total number of employees whose department code is HRD.**

```
[tybsc308@linserver tybsc308]$ grep -c HRD empdata308
2
```

**E) Create a file student<saetno> with the following fields:**

```
[tybsc308@linserver tybsc308]$ cat Student308
S01:Shweta:Q11:5:500
S02:Megha:Q12:1:400
S03:Milu:Q13:3:200
S04:Anu:Q11:2:100
S05:Ankit:Q15:4:308
```

**(i) Display the details of student in order of their name ignoring case.**

```
[tybsc308@linserver tybsc308]$ sort -i -t":" +1 -2
Student308
S05:Ankit:Q15:4:308
S04:Anu:Q11:2:100
S02:Megha:Q12:1:400
S03:Milu:Q13:3:200
S01:Shweta:Q11:5:500
```

**(ii) Display the details of student whose number of modules is greater than 3.**

```
[tybsc308@linserver tybsc308]$ grep ':[4-5]:' Student308
S01:Shweta:Q11:5:500
S05:Ankit:Q15:4:308
```

**(iii) Store the list of first 5 rank holders in merit<saetno> file.**

```
[tybsc308@linserver tybsc308]$ sort -t":" -r +4
Student308 | head -n 5 > merit2
89
[tybsc308@linserver tybsc308]$ cat merit308
S01:Shweta:Q11:5:500
S02:Megha:Q12:1:400
S05:Ankit:Q15:4:308
S03:Milu:Q13:3:200
S06:Anu:Q12:2:100
```

**(iv) Count number of students in Batch Q13.**

```
[tybsc308@linserver tybsc308]$ grep -c Q13 Student308
1
```

**(v) Display the list of students with same names.**

```
[tybsc308@linserver tybsc308]$ grep Anu Student308
S04:Anu:Q11:2:100
S06:Anu:Q12:2:100
```



## Practical 13

### LINUX: ADVANCED SHELL PROGRAMMING I

#### 1. To find the sum and product of integers.

```
[tybsc308@linserver tybsc308]$ cat calc
a=30
b=20
expr $a + $b
expr $a \* $b
echo
```

#### Output

```
[tybsc308@linserver tybsc308]$ bash calc
50
600
```

#### 2. To read the basic salary and find the gross salary.

```
[tybsc308@linserver tybsc308]$ cat pract7.3
echo Enter your basic salary
read basic
echo da
da= expr \( $basic \* 20 / 100 \)
echo hra
hra= expr \( $basic \* 30 / 100 \)
echo Taxes
Taxes= expr \( $basic \* 20 / 100 \)
echo Your gross salary is
expr $basic + \( $basic \* 20 / 100 \) + \( $basic \* 30 / 100 \)
\($basic \* 20 / 100 \)
```

#### Output

```
[tybsc308@linserver tybsc308]$ bash pract7.3
Enter your basic salary
1000
da
200
hra
308
Taxes
200
Your gross salary is
1308
```

**3. To check whether the file with the name entered exists or not.**

```
[tybsc308@linserver tybsc308]$ vi fileexists
echo enter filename
read filename
if [ -s $filename ]
then
    echo file exists
else
    echo does not exists
fi
```

**Output**

```
[tybsc308@linserver tybsc308]$ bash fileexists
enter filename
greenapple
file exists
```

**4. To compare the two strings.**

```
[tybsc308@linserver tybsc308]$ cat stringcompare
echo enter a string1:
read string1
echo enter another string2:
read string2
if [ $string1 = $string2 ]
then
    echo strings match
else
    echo strings do not match
fi
```

**Output**

```
[tybsc308@linserver tybsc308]$ bash stringcompare
enter a string1:
Unix
enter another string2:
Unix
strings match
```

**5. To check whether the file has a permission to write.**

```
[tybsc308@linserver tybsc308]$ cat filepermission
echo enter an existing file name
read filename
if [ -w $filename ]
then
    echo yes, the file has a write permission
else
    echo no write permission
fi
```

### Output

```
[tybsc308@linserver tybsc308]$ bash filepermission
enter an existing file name
greenapple
no write permission
```

### **6. To give grades using multiple if's.**

```
[tybsc308@linserver tybsc308]$ cat grades
echo enter your marks
read marks
if [ $marks -ge 75 -a $marks -lt 100 ]
then
    echo Your grade is A
elif [ $marks -ge 60 -a $marks -lt 75 ]
then
    echo Your grade is B
elif [ $marks -ge 50 -a $marks -lt 60 ]
then
    echo Your grade is C
elif [ $marks -ge 35 -a $marks -lt 50 ]
then
    echo Your grade is Pass
elif [ $marks -gt 0 -a $marks -lt 35 ]
then
    echo You have failed
elif [ $marks -gt 100 ]
then
    echo enter below 100
elif [ $marks -lt 0 ]
then
    echo Invalid entry
fi
```

### Output

```
[tybsc308@linserver tybsc308]$ bash grades
enter your marks
158
enter below 100
```

### **7. To check whether the number is +ve or -ve using if...elif.**

```
[tybsc308@linserver tybsc308]$ cat 10nos
echo enter a number
read a
if [ $a -lt 0 ]
then
    echo $a is negative
elif [ $a -gt 0 ]
```

```

then
    echo $a is positive
else
    echo number is neither positive nor negative
fi

```

#### **Output**

```

[tybsc308@linserver tybsc308]$ bash 10nos
enter a number
55
55 is positive

```

#### **8. To print the day of week using case ...in**

```

[tybsc308@linserver tybsc308]$ cat pract7.9
echo enter a value for a day
read d
case $d in
1) echo "Monday"
;;
2) echo "Tuesday"
;;
3) echo "Wednesday"
;;
4) echo "Thursday"
;;
5) echo "Friday"
;;
6) echo "Saturday"
;;
7) echo "Sunday"
;;
*) echo "Invalid Day Number"
esac

```

#### **Output**

```

[tybsc308@linserver tybsc308]$ bash pract7.9
enter a value for a day
6
Saturday

```

#### **9. To find the type of the character entered.**

```

[tybsc308@linserver tybsc308]$ cat pract7.10
echo enter the character
read char
case $char in
[a-z] )echo "You have entered small case"
;;
[A-Z] )echo "You have entered capital case"

```

```
;;
[0-9] )echo "You have entered number"
;;
?)echo "You have entered a special character"
;;
*)echo "You have entered more the one character"
;;
esac
```

#### **Output**

```
[tybsc308@linserver tybsc308]$ bash pract7.10
enter the character
a
You have entered small case
```

#### **10. To find the pattern of the string entered using case...in.**

```
[tybsc308@linserver tybsc308]$ cat pract7.11
echo enter the string
read s
case $s in
[aeiou]*)echo "The String begins with a small case vowel"
;;
[AEIOU]*)echo "The String begins with a capital case vowel"
;;
[0-9]*)echo "The String begins with a digit"
;;
*[0-9])echo "The String ends with a digit"
;;
?????)echo "You entered a five letter word"
;;
esac
```

#### **Output**

```
[tybsc308@linserver tybsc308]$ bash pract7.11
enter the string
i
The String begins with a small case vowel
```

**Write a shell script to generate prime series up to N.**

**Write down shell script in the answer sheet, execute it and show it to the examiner.**

```
[tybsc308@linserver tybsc308]$ cat pract8.11.c
#include<stdio.h>
#include<math.h>
int main(void)
{int no,i,j,prime=1,m;
printf("enter a number: ");
scanf("%d",&no);
if (no==1)
printf("1 is a prime");
if (no==2)
printf("2 is a prime number");
if (no%2==0)
    prime=0;
else
{
    for(i=3;i<=no;i=i+2)
    { prime=1;
        for(m=3;m<=(no/2);m=m+2)
        {
            if(no%i==0)
            {
                prime=0;
                break;
            }
        }
        if (prime==1)
            printf("\n%d is a prime number",i);
    }
}
}
```

### Output

```
[tybsc308@linserver tybsc308]$ gcc -o primeo.out pract8.11.c
[tybsc308@linserver tybsc308]$ ./primeo.out
Enter the no 11
3
5
7
```

---

## **Practical 14**

### **LINUX: EDITORS IN LINUX: SED EDITOR**

**Create a file with five records with the name fsed1 and do the following using sed command:**

```
[tybsc308@linserver tybsc308]$ cat fsed1
This is file fsed1
this is cat command.
hello.
hi.
tybsc.
```

**1) Display first three lines**

```
[tybsc308@linserver tybsc308]$ sed -n '1,3p' fsed1
This is file fsed1
this is cat command.
hello.
```

**2) Display the last line**

```
[tybsc308@linserver tybsc308]$ sed -n '$p' fsed1
tybsc.
```

**3) Display the third and fourth line**

```
[tybsc308@linserver tybsc308]$ sed -n '3,4p' fsed1
hello.
hi.
```

**4) Insert two more records and save the new file as newwed**

```
[tybsc308@linserver tybsc308]$ sed '2i\This is the 6th line\'
fsed1 > newwed
```

```
[tybsc308@linserver tybsc308]$ cat newwed
This is file fsed1
This is the 6th line
this is cat command.
hello.
hi.
tybsc.
```

```
[tybsc308@linserver tybsc308]$ sed '3i\This is the 3th line\'
fsed1 > newwed
```

```
[tybsc308@linserver tybsc308]$ cat newwed
This is file fsed1
this is cat command.
```

```
This is the 3th line
hello.
hi.
tybsc.
```

**5) Delete the last two records from the file newsed**

```
[tybsc308@linserver tybsc308]$ cat newsed
1
2
3
4
5
6
7
[tybsc308@linserver tybsc308]$ sed '$d' newsed | sed '$d'
1
2
3
4
5
```

**6) Link to the file :ln**

**A) Create two files with some contents with name ln1.**

```
[tybsc308@linserver tybsc308]$ vi ln1
[tybsc308@linserver tybsc308]$ cat ln1
This is file ln1
Practical no 10
```

**B) Copy ln1 to ln2.**

```
[tybsc308@linserver tybsc308]$ cp ln1 ln2
[tybsc308@linserver tybsc308]$ cat ln2
This is file ln1
Practical no 10
```

**C) Create a hard link to as "hardln1" and a soft link as "softln1" to ln1**

```
[tybsc308@linserver tybsc308]$ ln ln1 Hardln1
[tybsc308@linserver tybsc308]$ ln -s ln1 Softln1
[tybsc308@linserver tybsc308]$ ls
+1          emp          fspab          newsedclear    pract8.1
Softln1
10000       empdata308  fspac          pract           pract8.11.2
ss1
ashwini     fact         gre1           pract7.1        pract8.12.3
ss2
bdata308    fcmp1        gre2           pract7.10       pract8.14
ss3
```



check Stud308	fcmp1_308	green	pract7.11	pract8.3
checkgrade Student	fcmp2	<b>Hardln1</b>	pract7.12	pract8.5
date1	fod1	merit308	pract7.4	prime.out
xab				
dir1	foreg	month	pract7.5	product_308
xac				
dir2	fsed1	names	pract7.6	result
yellow				
dir3	fsp308	newBC	pract7.7	result1
doll	fspaa	newsed	pract7.9	Shweta308

**D) Construct the command to find the file permissions and inode numbers of the above three files.**

```
[tybsc308@linserver tybsc308]$ ls -li Hardln1 Softln1
1137795 Hardln1 1137795 Softln1
```

**E) Remove file ln1.**

```
[tybsc308@linserver tybsc308]$ rm ln1
```

**F) Type the contents of "hardln1" and "softln1".**

```
[tybsc308@linserver tybsc308]$ cat Hardln1
This is file ln1
Practical no 10
[tybsc308@linserver tybsc308]$ cat Softln1
This is file ln1
```

**G) What is your conclusion?**

Hard link 'Hardln1' shows the content as deleting the file 'ln1'.  
But Soft link 'Softln1' does not shows the content as original file is deleted.

---